

System and Method for Estimating Confidence and Implementing Metacognitive Abilities in Artificial Intelligence Systems

Technical Field

The present application relates generally to artificial intelligence, and in particular systems and methods for estimating confidence and implementing metacognitive abilities in artificial intelligence systems.

Background

Recent developments in artificial intelligence (AI) have led to the creation of increasingly sophisticated AI reasoners, capable of executing complex tasks and making autonomous decisions. Despite these advancements, these AI reasoners often lack metacognitive abilities essential for self-reflection, self-criticism, and self-directed optimization. This limitation restricts their ability to accurately estimate and communicate the confidence associated with their predictions, undermining the reliability and adaptability of the systems.

Moreover, AI reasoners usually function in an opaque manner, offering minimal insight into their decision-making processes and the reasoning behind their outputs. This lack of transparency may compromise the validation of the AI reasoners' conclusions and raise concerns about their reliability and accountability. Additionally, the inability to effectively convey the uncertainties associated with their predictions may result in outputs that are perceived as either overly confident or insufficiently assured, further reducing trust in these systems.

Another challenge is the restricted capacity of AI reasoners for continuous learning and autonomous enhancement. Although existing methods like reinforcement learning and transfer learning facilitate some degree of adaptation and performance improvement, they predominantly rely on external feedback and predefined objectives. The absence of inherent capabilities for self-assessment and targeted improvement limits the development of truly autonomous AI reasoners.

Therefore, it is desirable to provide a system and method for estimating confidence and implementing metacognitive abilities in AI systems to address the disadvantages or limitations of the existing technologies or, at the very least, provide the public with a useful alternative.

Summary

Embodiments herein provide new and useful systems and methods involving confidence estimation, metacognitive processing, and multi-agent orchestration in artificial intelligence systems.

In broad terms, the present disclosure proposes a method for generating a confidence score specific to an output from an artificial intelligence system including processing an input query by a language model to obtain an initial output, computing a plurality of confidence estimates for the initial output, wherein each confidence estimate is computed based on a distinct computational technique, and aggregating the confidence estimates based on a game-theoretical approach to generate an aggregated confidence score for the initial output.

In embodiments, the method further includes applying the aggregated confidence score to adjust parameters of an artificial intelligence system.

In implementations, the adjusted parameters of the artificial intelligence system are based on model performance metrics corresponding to a level of confidence determined by the aggregated confidence score.

In implementations, the game-theoretic approach is derived from a Nash Embedding Theorem, which optimizes the aggregation of the confidence estimates by evaluating an interdependency among the plurality of confidence estimates.

In embodiments, the distinct computational technique is selected from a group comprising: verbalized confidence assessments, token probability analysis, prompt entropy measurements, semantic output clustering, self-consistency diagnostics, and hidden state divergence metrics.

In embodiments, the method further includes using a structured data format to encode the confidence estimates and the aggregated confidence score, wherein the format is compatible with a standardized metadata representation.

In implementations the game-theoretical approach is based on feedback derived from system performance to enhance future confidence estimations.

In embodiments, the distinct computational technique includes calculating token probabilities by extracting SoftMax probabilities for each token from an output distribution of the language model and calculating an overall sequence probability as a product of the token probabilities.

In implementations, the distinct computational technique includes determining a prompt entropy by prompting the AI system with multiple label options, calculating an entropy of a

resulting probability distribution corresponding to the label options, and mapping entropy values to the confidence estimates based on a predetermined function.

In implementations, the distinct computational technique includes analysing a verbalized confidence by pattern matching to identify statements of confidence in an output of an AI system and assigning numerical scores to the identified statements based on a predefined mapping.

In embodiments, the distinct computational technique includes clustering semantic outputs by generating a plurality of output sequences, computing semantic embeddings for the output sequences, and using metrics to quantify a clustering quality.

The present disclosure further proposes a system for information processing, including a data acquisition module configured to receive feedback corresponding to one or more outputs generated by a language model and a cognitive reasoning module configured to evaluate the reasoning process of the language model, emulate cognitive functions including metacognitive processes, and generate an assessment based on an analysis of the received feedback. The assessment further includes classifying the one or more outputs into components, assigning quality scores for each component, and identifying an improvement corresponding to the one or more outputs. The system further includes a process adjustment module coupled to the cognitive reasoning module, configured to adjust the reasoning process of the language model based on the assessment and a refinement module coupled to the process adjustment module, configured to iteratively refine the reasoning process based on subsequent updates to the generated assessment until a performance threshold is met.

The present disclosure further proposes a method for information processing, including receiving feedback corresponding to one or more outputs generated by a language model and using a reasoning module to evaluate the reasoning process of the language model, emulate cognitive functions including metacognitive processes, and generate an assessment based on an analysis of the received feedback. The assessment includes classifying the one or more outputs into components, assigning quality scores for each component, and identifying an improvement corresponding to the one or more outputs. The method further includes adjusting the reasoning process of the language model based on the assessment and iteratively refining the reasoning process based on subsequent updates to the generated assessment until a performance threshold is met.

In embodiments, the classifying of the one or more outputs into components includes applying a modularized analysis that quantifies evaluation metrics for each component based on relevance, coherence, factual accuracy, and completeness

5 In implementations, the assigning of quality scores for each component utilizes techniques such as token probabilities, prompt entropy, and semantic output clustering to calibrate the confidence scoring.

In embodiments, the identifying of an improvement includes employing a counterfactual reasoning process to evaluate potential alternative outcomes and their impacts on the reasoning process of the language model.

10 In implementations, the reasoning module further comprises a sub-module for generating detailed logs of each iteration in the refinement process, which includes recording changes to reasoning strategies and their effects on output quality.

The present disclosure further proposes a system for coordinating operations, including a model configuration module configured to define a structured framework of a multi-agent
15 system, wherein each component of the multi-agent system is assigned tasks related to processing outputs generated by an artificial intelligence (AI) system and an agent coordination module configured to manage interactions and synchronize data flow among agents of the system based on roles or dependencies corresponding to each agent within the structured framework and utilize mechanisms for coordination and synchronization of the
20 agents of the system.

In embodiments, the model configuration module is further configured to utilize Petri nets to define the structured framework, wherein the Petri nets specify the roles or dependencies of each agent within the multi-agent system.

25 In embodiments, the agent coordination module utilizes a conflict resolution strategy to manage data flow among the agents when multiple agents access data resources simultaneously.

In implementations, each agent within the multi-agent system is configured to generate and send feedback regarding their task execution to the model configuration module, which uses the feedback to refine task assignments in subsequent operations.

30 In implementations, the model configuration module assigns the tasks to agents based on a dynamic assessment of the operational load and performance metrics of each agent.

The present disclosure further proposes a method for coordinating operations within a multi-agent system, including implementing a structured framework of a multi-agent system, wherein each component of the multi-agent system is assigned specific tasks related to processing outputs generated by an artificial intelligence (AI) system. The method for coordinating operations further includes managing interactions and synchronizing data flow among agents of the multi-agent system based on roles or dependencies corresponding to each agent within the structured framework and utilizing mechanisms for coordination and synchronization of the agents of the system.

The above description is provided as an overview of some implementations of the present disclosure. Further description of those implementations, and other implementations, are described in more detail below.

Brief Description of the Drawings:

Embodiments of the invention will now be explained for the sake of example only, with reference to the following figures in which:

FIG. 1 is a functional block diagram illustrating an example of an information processing system for enhancing decision-making within an artificial intelligence framework, according to an embodiment herein

FIG. 2 is a flowchart illustrating an example process for generating an aggregated confidence score in an artificial intelligence system, according to an embodiment herein.

FIG. 3 illustrates a flowchart illustrating operations for managing and integrating multiple confidence estimates in an artificial intelligence system using a game-theoretic approach, according to an embodiment herein.

FIG. 4 is a flowchart illustrating an example process for managing a multi-agent system within an artificial intelligence system, according to an embodiment herein.

FIG. 5 is a block diagram illustrating an example computer system which may be configured to implement the systems and methods as disclosed herein.

Detailed Description

Embodiments will now be discussed with reference to the accompanying FIGs, which depict one or more exemplary embodiments. These embodiments are described in sufficient detail to enable those skilled in the art to practice the embodiments and it is to be understood that mechanical, logical, and other changes may be made without departing from the scope of

the embodiments. Therefore, embodiments may be implemented in many different forms and should not be construed as limited to the embodiments set forth herein, shown in the FIGs, and/or described below.

As used in this disclosure, the terms “component,” “module,” “system,” “apparatus,”

“interface,” or the like are generally intended to refer to a computer-related entity, either hardware, a combination of hardware and software, software, or software in execution. For example, a component or a module may be, but is not limited to being, a process running on a processor, a processor, an object, an executable, a thread of execution, a program, and/or a computer. By way of illustration, both an application running on a controller and the controller can be a component or a module. One or more components/modules may reside within a process and/or thread of execution and a component may be localized on one computer and/or distributed between two or more computers.

Unless otherwise defined, all terms (including technical and scientific terms) used herein are to be interpreted as is customary in the art. It will be further understood that terms in common usage should also be interpreted as is customary in the relevant art.

A system and method for implementing neuromorphic reasoning called COGNIGEN-AX (Cognitive Optimization and Generative Enhancement through Introspective Governance and Explainable Adaptive Cross-domain Reasoning) is disclosed herein.

In embodiments, the COGNIGEN-AX system and method generates autonomous AI reasoners with advanced metacognitive abilities and confidence estimation techniques. COGNIGEN-AX may combine self-reflection, self-criticism, and self-directed optimization methods with confidence estimation capabilities to create AI systems that operate with autonomy, adaptability, and transparency.

In embodiments, COGNIGEN-AX provides a retrospective experience-driven iterative multi-model adaptive self-optimization approach. This approach enables AI reasoners to interpret evaluative feedback signals, generate self-reflective critiques, and dynamically adapt their knowledge models and inferential strategies based on insights. COGNIGEN-AX may incorporate an introspective performance analysis and self-enhancement methodology, which includes self-critique, evaluation, root cause analysis, and remediation planning.

These components may be seamlessly integrated into a workflow for continuous AI optimization through self-reflection, validation, feedback assimilation, and interventions.

In implementations, COGNIGEN-AX uses confidence estimation techniques to generate reliable and calibrated confidence scores for the outputs of an AI reasoner. These

techniques include methods for verbalized confidence, token probabilities, prompt entropy, semantic output clustering, and self-consistency diagnostics, along with a hidden state divergence metric. Integrating these techniques enhances the AI reasoner's evaluation and optimization processes, enabling more informed and self-aware decision-making

- 5 In implementations, COGNIGEN-AX employs confidence-based prompting strategies, such as chain-of-thought prompting, self-evaluating prompts, and Socratic questioning, to generate informative and self-aware predictions from the AI reasoner. These strategies enable the reasoner to provide step-by-step reasoning, self-evaluation, and guided self-questioning to achieve accurate and well-justified conclusions.
- 10 In order to combine the outputs of multiple reasoning modules or iterations, COGNIGEN-AX uses confidence-based prediction ensembling techniques, including weighted averaging of predictions and threshold voting. These techniques may prioritize reliable predictions and minimize the impact of uncertain outputs. Furthermore, COGNIGEN-AX may employ a game-theoretic approach, grounded in the Nash Embedding Theorem, to incorporate
- 15 confidence estimates derived from various sources and techniques within its workflow. This implementation considers the aggregation of confidence as a decentralized mechanism design issue, thereby allowing COGNIGEN-AX to acquire a principled and adaptable strategy for amalgamating confidence estimates in a self-adjusting manner.

- COGNIGEN-AX provides various advantages for enhancing overall AI functionality and
- 20 reliability. For example, the integration of advanced confidence estimation techniques substantially improves the precision and reliability of the AI reasoner's outputs. Moreover, the system's retrospective, experience-driven adaptive optimization approach, coupled with its introspective performance analysis, significantly boosts the AI's self-awareness and self-correction capabilities. This leads to more effective iterative refinement and ongoing
- 25 learning, driven by a robust, integrated workflow that continuously optimizes AI operations through external validation and user feedback. Additionally, the adoption of standardized data interchange formats and the inclusion of confidence estimation metadata increase the transparency and interpretability of the AI's decision-making processes, allowing
- COGNIGEN-AX to develop more reliable, self-aware, and adaptable AI reasoners across
- 30 various domains.

COGNIGEN-AX may incorporate advanced metacognitive abilities with sophisticated confidence estimation techniques within a comprehensive, cyclical workflow, enabling AI reasoners to systematically enhance functionality and self-regulate through continuous optimization and precise adaptation. COGNIGEN-AX may include the following features:

Integration of Metacognitive Abilities and Confidence Estimation Techniques: COGNIGEN-AX may combine advanced metacognitive abilities, such as self-reflection, self-criticism, and self-directed optimization, with sophisticated confidence estimation techniques. This integration enables AI reasoners to engage in structured self-improvement while generating reliable and well-calibrated confidence scores for their outputs.

Synergistic Workflow for Continuous AI Optimization: COGNIGEN-AX may incorporate a comprehensive workflow called ISOCLES (Iterative Self-Optimization through Continual Learning and Evaluation Synthesis), which includes REIMAS (Retrospective Experience-driven Iterative Multi-model Adaptive Self-optimization), IPASE (Introspective Performance Analysis and Self-Enhancement), and COGNATE (Composable Game-theoretic Nash-embedded Adaptive Techniques for Estimating Confidence) modules. This process provides continuous AI optimization through cyclic self-reflection, external validation, user feedback assimilation, and targeted developmental interventions, leading to a high degree of self-regulated autonomous analysis and refinement.

Comprehensive Confidence Estimation Suite: COGNIGEN-AX may include a suite of advanced confidence estimation techniques, including verbalized confidence, token probabilities, prompt entropy, semantic output clustering, self-consistency diagnostics, and a hidden state divergence metric (HSDM). These diverse and complementary techniques provide a nuanced evaluation of AI output confidence, enhancing the capabilities beyond existing methods.

Game-Theoretic Approach for Confidence Integration: COGNIGEN-AX may employ a game-theoretic approach based on the Nash Embedding Theorem for the optimal integration of confidence estimates from various sources. This approach considers confidence aggregation as a decentralized mechanism design problem, facilitating an adaptive and principled composition of confidence estimation techniques.

Strategies for Prompting Based on Confidence: The system may implement innovative prompting strategies such as chain-of-thought prompting, self-evaluating prompts, and Socratic questioning. These strategies improve upon traditional methods by enabling deeper introspective assessments of AI outputs, addressing gaps not covered in existing literature.

Modular and Hierarchical Multi-Agent Orchestration Using Petri Nets: COGNIGEN-AX may utilize modular and hierarchical multi-agent orchestration with executable Petri Net models that adhere to the ISO/IEC 15909 standard. This method ensures precise control over agent interactions and execution flow, providing a formal and standardized framework for multi-agent coordination.

Extension of ISO 11179 for Confidence Estimation Metadata: COGNIGEN-AX may extend the ISO 11179 metadata standard to include attributes specifically related to confidence estimation. This enhancement supports standardized representation and exchange of confidence-related data throughout the optimization workflow, improving interoperability, traceability, and interpretability of confidence estimation processes.

Overview of COGNIGEN-AX System and Method

In examples, COGNIGEN-AX comprises four modules—REIMAS (Retrospective Experience-driven Iterative Multi-model Agent Self-optimization), IPASE (Introspective Performance Analysis and Self-Enhancement), COGNATE (Composable Game-theoretic Nash-embedded Adaptive Techniques for Estimating Confidence), and ISOCLES (Iterative Self-Optimization through Continual Learning and Evaluation Synthesis). These modules enable artificial intelligence agents, including large language models and multimodal AI architectures, to function as self-improving autonomous reasoners with metacognitive abilities. By employing an iterative approach of self-orchestration, structured self-reflection, self-criticism, and self-directed optimization, these modules facilitate the continuous enhancement of the AI agents' capabilities.

FIG. 1 is a functional block diagram 100 illustrating an example of an information processing system for enhancing decision-making within an artificial intelligence framework, according to an embodiment herein. This diagram 100 illustrates the COGNIGEN-AX system's structured layout, highlighting the interactions among modules. In the example embodiment, these modules collaboratively function to estimate confidence levels and implement metacognitive abilities in AI systems, enhancing the system's decision-making capabilities as detailed herein

Data Acquisition Module 102: This module is responsible for the initial collection of data which may include user input, external data sources, or feedback from system outputs. It serves as the foundational layer that gathers necessary inputs for further processing and analysis within the system.

Cognitive Reasoning Module 104: Following data acquisition, the input data is forwarded to the cognitive reasoning module. This module processes the acquired data using various AI techniques to perform tasks such as initial data assessment, preliminary reasoning, and confidence estimation. It may use methods like natural language processing, machine learning algorithms, or other forms of data analysis to interpret and reason about the input data.

Process Adjustment Module 106: The outputs from the cognitive reasoning module are then passed to the process adjustment module. This module adjusts the AI system's processing parameters based on the cognitive assessments made earlier. Adjustments may include tuning algorithms, modifying decision thresholds, or recalibrating system settings to optimize performance according to the current task demands and data characteristics.

Refinement Module 108: The final module in the system is the refinement module. It takes the adjusted processes from the previous module and refines them further to enhance the AI system's decision-making capabilities. This could involve iterative enhancements, applying additional filters, or incorporating new data insights to refine the system's outputs.

Overall, the information processing system depicted in FIG. 1 showcases a structured approach to enhancing AI decision-making capabilities through a sequential processing framework that integrates data acquisition, cognitive reasoning, process adjustment, and refinement phases.

REIMAS

REIMAS (Retrospective Experience-driven Iterative Multi-model Agent Self-optimization) is a methodology that allows AI agents to interpret multi-format feedback on their outputs, generate self-reflective critiques, and dynamically adjust their knowledge models and inferential strategies based on accumulated insights. The steps in the REIMAS method are as follows:

1. The AI reasoning module generates an initial response to a given input query or task specification.

2. Obtaining multi-format feedback on the initial response from various sources, including user interactions, simulations, and external evaluations.

3. Analyzing the feedback to identify patterns, anomalies, and areas for improvement, and generating a self-reflective critique that highlights the strengths, weaknesses, and potential enhancements of the initial response.

4. Encoding the initial response, feedback, and self-reflective critique into a standardized format that complies with the ISO 11179 metadata registries standard. Storing the encoded artifacts in a knowledge repository as part of an auditable self-optimization trail.

5. Utilizing the accumulated insights from the self-optimization trail to dynamically adjust the AI agent's knowledge models and inferential strategies for future tasks.

IPASE:

IPASE (Introspective Performance Analysis and Self-Enhancement) is a methodology that provides protocols for conducting structured self-critiques, identifying areas for improvement, and proposing targeted remediation plans. The steps in the IPASE method are as follows:

- 5 1. Segmenting the initial response into distinct components or aspects.
2. Evaluating each component using predefined quality metrics, such as accuracy, coherence, completeness, and effectiveness.
3. Identifying limitations, errors, and areas for improvement in each component.
4. Conducting a root cause analysis of the identified issues using fault tree analysis and
10 fishbone diagrams.
5. Proposing targeted remediation or optimization strategies for each identified issue, prioritizing them based on impact and feasibility.
6. Generating a detailed self-critique report that includes the segmented analysis, quality metrics, identified issues, root causes, and remediation proposals.
- 15 7. Encoding the self-critique report into the ISO 11179-compliant format and storing it in the knowledge repository.
8. Integrating the remediation proposals into the AI agent's reasoning processes and knowledge models.

COGNATE

- 20 COGNATE (Composable Game-theoretic Nash-embedded Adaptive Techniques for Estimating Confidence) is a module that integrates a set of confidence estimation methods, including:
 1. VETTED (Verbalized confidence, Token probabilities, prompt Entropy, semantic output clustering, and self-consistency Diagnostics) methods enhanced with the Hidden State
25 Divergence Metric (HSDM).
 2. Confidence-based prediction ensembling techniques, such as weighted averaging of predictions and threshold voting.

3. Nash Embedding for confidence integration, which formulates confidence aggregation as a decentralized mechanism design problem, allows the system to learn a principled and adaptive strategy for combining confidence estimates.

The COGNIGEN-AX system employs a game-theoretic approach based on the Nash Embedding Theorem to optimally integrate confidence estimates from multiple techniques such as token probabilities, prompt entropy, and verbalized confidence.

Game Definition: In a non-cooperative game, each confidence estimation technique (e.g., token probabilities, prompt entropy) is treated as a self-interested agent. The goal of each agent is to maximize its accuracy while contributing to the overall performance of the ensemble.

Strategies: The strategy space for each agent consists of the possible mappings (from input to output concepts) and their associated confidence scores that the agent can produce.

Utility Functions: Each agent is assigned a utility function that rewards accurate predictions and penalizes incorrect ones. This incentivizes the agents to maximize their accuracy.

Additionally, the utility function includes a term that rewards agents for contributing to the ensemble's overall accuracy and calibration, promoting cooperation and specialization.

Payoff Matrix Construction: A payoff matrix encodes the incentives for each technique to specialize in different mapping categories based on their empirical accuracies. The mapping inputs are categorized based on complexity, target taxonomy size, or other relevant features.

The accuracy (or reward) obtained by each technique for each category is evaluated on a validation set and recorded in the payoff matrix.

Nash Equilibrium Computation: Given the payoff matrix, the Nash equilibrium of the game is computed using techniques like the Lemke-Howson algorithm or linear programming. The equilibrium solution represents the optimal strategy profile, a probability distribution over the confidence estimation techniques. This distribution indicates the proportion of mappings that should be assigned to each method based on their specialization.

Referee Introduction: A referee agent is introduced to coordinate the composition of techniques based on the computed Nash equilibrium. The referee assigns incoming mappings to the appropriate methods according to their assigned proportions in the equilibrium. It collects the individual confidence scores from each technique and combines them using a weighted aggregation scheme, where the equilibrium proportions determine the weights.

Dynamic Specialization: By leveraging the game-theoretic approach, confidence estimation techniques are incentivized to specialize in different input patterns dynamically. Techniques focus on mapping categories where they exhibit higher accuracy, leading to improved overall performance.

- 5 Ensemble Calibration: The referee agent learns to compose the techniques optimally into a well-calibrated ensemble. The final confidence estimates are reliable and well-aligned with the actual mapping accuracies, as the game-theoretic model encourages specialization and cooperation among the techniques.

The key conditions and assumptions of this game-theoretic model include:

- 10 The techniques are treated as rational agents aiming to maximize their utility (accuracy and ensemble contribution).

The mapping categories are distinguishable based on relevant features, and the empirical accuracies of techniques can be estimated for each category.

- 15 The Nash Embedding Theorem guarantees the existence of at least one Nash equilibrium, which states that every game with a non-empty, compact strategy space and continuous utility functions has a Nash equilibrium.

- By formulating the confidence integration problem as a non-cooperative game and leveraging the Nash Embedding Theorem, the COGNIGEN-AX system achieves an adaptive and principled composition of confidence estimation techniques, ensuring reliable and well-calibrated confidence estimates for the AI reasoners' outputs.
- 20

Example:

Game Definition:

Confidence estimation techniques like Verbalized Confidence, Token Probabilities, Prompt Entropy, and Self-Consistency are treated as self-interested agents.

- 25 Their goal is to maximize accuracy in estimating confidence for mapping "% Premium method ranking" to metadata attributes like Concept, Description, etc., while contributing to the ensemble's overall performance.

Strategies:

- Each agent's strategy space consists of possible mappings from the input "% Premium method ranking" to output attributes like:
- 30

Verbalized Confidence: "Premium method ranking" (Concept), "Ranking of the percentage premium method..." (Description), Confidence Score: 0.9

Token Probabilities: "Premium method ranking" (Concept), "Indicates the relative importance..." (Meaning), Confidence Score: 0.8...

5 Utility Functions:

Utility function for Token Probabilities agent: $U(\text{Token Probabilities}) = 0.5 * (\text{Mapping Accuracy}) + 0.3 * (\text{Confidence Calibration}) + 0.2 * (\text{Contribution to Ensemble Accuracy})$

Rewards accurate mapping of "% Premium method ranking," well-calibrated confidence scores, and contribution to overall ensemble accuracy.

10 Payoff Matrix:

Categorize mapping inputs based on complexity (e.g., length, insurance terminology), target taxonomy size (e.g., number of metadata attributes), or data quality.

Evaluate the accuracy of each technique per category on a validation set of insurance concepts.

15 Record each technique and category's payoffs (rewards) in the payoff matrix.

Nash Equilibrium:

Compute the Nash equilibrium of the game using the payoff matrix, e.g., via the Lemke-Howson algorithm.

20 Equilibrium could be: Token Probabilities: 0.4, Verbalized Confidence: 0.3, Prompt Entropy: 0.2, Self-Consistency: 0.1

Indicates the proportions for assigning mappings to each technique based on their specializations.

Referee:

25 The referee assigns the "% Premium method ranking" mapping based on the equilibrium proportions (e.g., 40% to Token Probabilities, 30% to Verbalized Confidence, etc.).

Collects confidence scores from each technique and combines them using a weighted average, with equilibrium proportions as weights.

Specialization and Calibration:

Techniques specialize in input patterns that exhibit higher accuracy (e.g., Token Probabilities for complex insurance terms, Verbalized Confidence for more straightforward concepts).

The referee composes techniques optimally into a well-calibrated ensemble for mapping insurance concepts like "% Premium method ranking," ensuring reliable and accurate confidence estimates aligned with actual mapping performance.

ISOCLES

ISOCLES (Iterative Self-Optimization through Continual Learning and Evaluation Synthesis) is a workflow that combines the REIMAS, IPASE, and COGNATE modules into a system for continuous AI optimization. ISOCLES uses a modular and hierarchical multi-agent orchestration engine that utilizes executable Petri Net models with clear state specifications to control the execution flow and state transitions of different agents. The definitions and behavior of each agent may be stored within reusable and version-able Petri Net modules, maintained under strict version control following industry-standard practices for branching and merging.

The ISOCLES workflow may be executed by a top-level Petri Net that orchestrates the following agents in sequence, considering parallelization opportunities, realistic timing constraints, and synchronization mechanisms. The Petri Net models may adhere to the ISO/IEC 15909 standard, defining all possible states an agent can transition into, including error/exception states driven by appropriate token-based preconditions and guards. High-level tokens with rich data payloads, following the ISO 11179 metadata standard, may be used to pass inputs/outputs between agents. If any agent transitions to a "Fail" state, the top-level net halts execution, performing necessary process cleanup actions and rollback operations. Periodic analysis may be conducted to ensure desired orchestration properties, such as liveness, boundedness, and persistence, and to verify the absence of deadlocks and livelocks. The ISOCLES workflow ensures that the AI agent continuously learns from its experiences, refines its reasoning capabilities, and adapts to new challenges over time. By integrating the REIMAS, IPASE, and COGNATE modules using a modular and hierarchical multi-agent orchestration approach, ISOCLES enables the AI agent to develop a comprehensive understanding of its strengths, limitations, and areas for improvement, and to take proactive steps toward self-enhancement.

Design of COGNIGEN-AX

Highly self-regulating autonomous AI reasoners:

The development of COGNIGEN-AX involves the use of AI reasoners with a high level of autonomy and self-regulation. This achievement encompasses several important objectives. Firstly, COGNIGEN-AX enhances the metacognitive capabilities of AI reasoners, enabling them to engage in self-reflection, self-monitoring, and self-control. By incorporating advanced metacognitive processes, AI reasoners can have a better understanding of their reasoning patterns, identify potential errors or biases, and adjust their strategies accordingly. Secondly, COGNIGEN-AX incorporates advanced confidence estimation techniques, allowing AI reasoners to accurately assess the reliability and uncertainty of their outputs. This enables them to effectively communicate confidence levels and make more informed decisions. Lastly, COGNIGEN-AX facilitates self-reflection, self-criticism, and self-directed optimization, empowering AI reasoners to continuously evaluate and enhance their performance without extensive human intervention.

Improve reliability and transparency of AI outputs:

Another important objective of COGNIGEN-AX is to enhance the reliability and transparency of AI-generated outputs. This objective encompasses several goals aimed at improving the accuracy, consistency, and interpretability of AI decisions and predictions. COGNIGEN-AX has been designed with techniques and mechanisms that enable AI reasoners to produce more precise and consistent outputs across various tasks and domains. The system and methods effectively reduce errors and discrepancies in AI-generated results by incorporating robust error detection and correction methods. Moreover, the system and methods place strong emphasis on providing clear explanations and justifications for AI decisions. By developing advanced techniques for generating explanations and user interfaces, the system and methods make the AI reasoning process more transparent and understandable for human users. In addition, the system and methods quantify and communicate the uncertainty in AI predictions, ensuring that users understand the confidence levels associated with each output.

Continuous learning and adaptation

COGNIGEN-AX allows AI systems to learn from their experiences, refine their models based on feedback, and adapt to new situations and challenges. COGNIGEN-AX facilitates the iterative refinement of AI models by incorporating mechanisms for integrating user feedback, domain expertise, and real-world experiences into the learning process. COGNIGEN-AX continuously updates and optimizes AI models based on these inputs, improving the performance and generalization capabilities of AI reasoners over time. Additionally, it includes techniques for integrating new knowledge and skills into existing AI models, enabling them to expand their capabilities and handle new tasks. Lastly, the system and

methods optimize AI performance in dynamic and evolving environments by developing adaptive learning strategies and flexible architectures that adjust to changing conditions and requirements.

Metacognitive reasoning

- 5 COGNIGEN-AX utilizes advanced metacognitive reasoning capabilities to create highly automated AI reasoners. This involves equipping AI systems with the necessary tools and mechanisms for self-reflection, introspection, and self-monitoring. With COGNIGEN-AX, AI reasoners are able to examine their reasoning processes, mental states, and decision-making patterns through self-reflection and introspection mechanisms. These mechanisms
- 10 allow the AI systems to identify any potential errors, biases, or limitations in their reasoning and take appropriate corrective actions. Additionally, COGNIGEN-AX incorporates self-criticism and error detection algorithms that automatically detect and highlight any inconsistencies, anomalies, or mistakes in the outputs generated by the AI. These algorithms deter against inaccurate or unreliable results. Furthermore, COGNIGEN-AX
- 15 offers self-optimization and adaptation strategies that enable AI reasoners to adjust their parameters, models, or learning approaches based on performance evaluations and feedback received.

The COGNIGEN-AX system implements metacognitive abilities and self-awareness through several key components and methodologies:

- 20 The REIMAS (Retrospective Experience-driven Iterative Multi-model Adaptive Self-optimization) methodology enables AI reasoners to interpret feedback signals, generate self-reflective critiques, and dynamically adapt their knowledge models and reasoning strategies based on insights gained from experience.

- The IPASE (Introspective Performance Analysis and Self-Enhancement) methodology
- 25 provides rigorous protocols for modularized self-critique, quantitative evaluation, root cause analysis, and targeted remediation planning. Key agents/algorithms involved include:

MACQ: Conducts modularized analysis by breaking down the AI output into components like relevance, coherence, factual accuracy, and completeness. It quantifies evaluation metrics for each component to assess quality objectively.

- 30 RAINS (Root Cause Analysis and Insight Synthesis): Performs structured root cause analysis using techniques like the 5 Whys methodology, fishbone diagrams, and Pareto analysis to identify underlying factors contributing to limitations or errors. It constructs causal graphs visualizing the relationships between factors.

MIRROR: Facilitates textual self-reflection by generating comprehensive self-analyses identifying strengths, weaknesses, and areas for improvement in the outputs. It employs techniques like multi-perspective analysis, reasoning error identification, and counterfactual reasoning for introspection.

- 5 The CERES (Contextual Encoding and Reflective Experience Structuring) agent captures the full context of the AI's experience, including prompts, outputs, feedback, and self-reflections in a structured ISO 11179 metadata representation. This structured experience encoding enables iterative refinement based on accumulated insights.

- 10 The ALMA (Adaptive Learning and Model Augmentation) agent analyzes the accumulated experience logs, identifies patterns, formulates strategic reinforcement plans to expand knowledge bases, improve output quality, and enhance metacognitive capabilities through techniques like case-based reasoning and ontology learning.

Example of Metacognitive Reasoning:

- 15 Record: SARB:ei93:RTG11,% Premium method ranking,ei93, RTG11, Hierarchy for ranking of methods used in provisions assumptions (enumeration), Ratings,, "Method no.1 in ranking, Method no.2 in ranking,Method no.3 in ranking,Method no.4 in ranking,Method no.5 in ranking,Method no.6 in ranking,Method no.7 in ranking,Method no.8 in ranking,Method no.9 in ranking"

Initial Output Generation:

- 20 "SARB:ei93:RTG11" | "% Premium method ranking" | "Premium method ranking" | "Ranking of the percentage premium method among various methods used for calculating provisions assumptions." | "Indicates the relative importance or position of the percentage premium method within the hierarchy of methods employed for estimating provisions." | "Used in the context of rating assessments and determining the appropriate assumptions for calculating provisions." | "% Premium method ranking is part of the hierarchy for ranking methods used in provisions assumptions (enumeration) in the Ratings domain." | "Medium" | "4" | "High"
- 25

MIRROR Self-Reflective Critique:

- 30 Strength: The output effectively captures the essence of the concept by ranking the percentage premium method among various methods used for calculating provisions assumptions. It conveys that the concept relates to the relative importance or position of the percentage premium method within a hierarchy of provisioning methods. The output also

specifies the context in which this ranking is used, namely in rating assessments and determining appropriate assumptions for calculating provisions.

Weakness: However, the description needs more details on how the percentage premium method works and why its ranking matters. It needs to provide more information on the

- 5 specific calculation or assumptions involved in the percentage premium method, making it difficult for users to understand its underlying mechanics. Additionally, the description does not explain the implications or consequences of the method's ranking, such as how it affects the accuracy or reliability of provision estimates.

- Improvement: To address these weaknesses, the output could be improved by providing a
10 more comprehensive explanation of the percentage premium method and its implications for provisioning. This could include:

This is a brief overview of how the percentage premium method calculates provisions, such as estimating provisions as a fixed percentage of premiums earned and assuming a stable loss ratio over time.

- 15 The key assumptions or limitations of the percentage premium method influence its ranking, such as its reliance on historical loss ratios and sensitivity to changes in premium volume or claim frequencies.

- The implications of the method's ranking for provisioning accuracy and decision-making include how a higher ranking might indicate more excellent reliability or broader acceptance
20 of the method within the industry.

- By incorporating these additional details, the output would provide a more comprehensive and informative description of the "% Premium method ranking" concept, enabling users to understand better its mechanics, significance, and potential impact on provisioning practices. The MIRROR agent's self-reflective critique highlights the strengths of the initial output in
25 capturing the essence of the concept and its context while also identifying the weaknesses in the lack of details on the method's workings and implications. By proposing targeted improvements, including an overview of the calculation, key assumptions, and impact of the ranking, the MIRROR agent demonstrates its ability to analyze the output critically and suggest enhancements to provide a more comprehensive and valuable description of the
30 concept.

MACQ Agent's Modularized Analysis:

Relevance: High (90%)

The "% Premium method ranking" concept is highly relevant to the insurance provisioning process. Provisioning is a critical aspect of insurance operations, as it involves estimating the funds required to cover future claims and expenses. The percentage premium method is one of the critical methods used in calculating these provisions, and its ranking directly reflects its importance and reliability within the provisioning process. The output captures this relevance by stating that the concept is part of the "Hierarchy for ranking of methods used in provisions assumptions" and is used in the context of "rating assessments and determining the appropriate assumptions for calculating provisions." Therefore, the MACQ agent assigns a high relevance score of 90% to the output.

10 Coherence: Medium (70%)

The output concisely describes the "% Premium method ranking" concept, clearly stating its definition and purpose within the insurance provisioning context. However, the description needs some connective details that would enhance its coherence. For example, it does not explain how the ranking of the percentage premium method relates to the other methods in the hierarchy or how it affects the overall provisioning process. The lack of these connecting details makes the description somewhat disjointed and may hinder a complete understanding of the concept's significance. As a result, the MACQ agent assigns a medium coherence score of 70% to the output.

Factual Accuracy: High (95%)

20 The information provided in the output aligns well with the knowledge of the insurance domain. The percentage premium method is a commonly used approach for calculating provisions, and its ranking is a crucial factor in assessing the reliability and appropriateness of provisioning assumptions. The output accurately states that the concept is part of the "Hierarchy for ranking methods used in provisions assumptions" and is utilized in "rating assessments and determining the appropriate assumptions for calculating provisions." This information is consistent with the industry practices and understanding of the provisioning process. Consequently, the MACQ agent assigns a high factual accuracy score of 95% to the output.

Completeness: Low (60%)

30 While the output captures the essential aspects of the "% Premium method ranking" concept, it misses some key details that would provide a more comprehensive understanding. The description must explain how the percentage premium method calculates provisions by estimating them as a fixed percentage of premiums earned. It also

requires more information on the specific assumptions or limitations of the technique that influence its ranking, such as its reliance on historical loss ratios or sensitivity to changes in premium volume. Moreover, the output does not elaborate on the implications of the method's ranking for provisioning accuracy and decision-making. These additional details are necessary for the description to feel complete and fully convey the significance and nuances of the concept. Therefore, the MACQ agent assigns the output a low completeness score of 60%.

In this example, the MACQ agent's analysis highlights that while the output is highly relevant and factually accurate, it lacks some coherence and completeness. The high relevance score of 90% indicates that the concept is closely tied to the insurance provisioning process. The high factual accuracy score of 95% suggests that the information aligns with the domain knowledge. However, the medium coherence score of 70% points out the lack of connective details that would improve the flow and understanding of the description. The low completeness score of 60% emphasizes the missing vital aspects of the percentage premium method that would provide a more comprehensive picture of the concept.

RAINS Agent's Root Cause Analysis:

The RAINS agent performs a root cause analysis:

5 Whys Analysis: "Why is the description incomplete?" -> "Why does it lack details on the percentage premium method?" -> "Why is the source record not providing enough context?"

Fishbone Diagram: Factors contributing to the incompleteness include limited source data, domain knowledge gaps, and mapping rules focused on brevity.

Causal Graph: The incomplete description is caused by the need for more contextual details in the source record.

5 Whys Analysis:

The RAINS agent begins the root cause analysis by applying the 5 Whys technique to investigate the underlying reasons for the incomplete description of the "% Premium method ranking" concept.

Why is the description incomplete? The description needs more details on how the percentage premium method works and its implications.

Why does it lack details on the percentage premium method? Answer: The description does not provide information on the method's specific calculations, assumptions, or limitations.

Why is the source record not providing enough context? - Answer: The source record focuses on the ranking aspect and does not include additional details about the method.

The 5 Whys analysis reveals that the incomplete description can be traced back to the limited contextual information provided in the source record. The source record primarily focuses on the ranking of the percentage premium method within the hierarchy of provisioning methods, but it needs to offer more details on the method's mechanics, assumptions, or implications.

Fishbone Diagram:

Next, the RAINS agent creates a fishbone diagram to identify and categorize the factors contributing to the incomplete description.

Factor 1: Limited source data: The source record needs to include comprehensive information about the percentage premium method. It also requires more details on the method's calculations, assumptions, and limitations, which hinders a complete description.

Factor 2: Domain knowledge gaps: The mapping process may need access to the full insurance provisioning domain knowledge. Gaps in understanding the intricacies of the percentage premium method and its implications can lead to an incomplete description.

Factor 3: Mapping rules focused on brevity: The mapping rules may prioritize concise descriptions over comprehensive explanations. An emphasis on brevity can result in omitting key details and nuances about the percentage premium method. The fishbone diagram helps categorize the contributing factors into three main areas: limited source data, domain knowledge gaps, and mapping rules focused on brevity. This categorization provides a structured view of the various influences leading to the incomplete description.

Causal Graph:

Finally, the RAINS agent constructs a causal graph to visualize the relationships between the identified factors and the incomplete description.

The lack of contextual details in the source record is the leading cause of the incomplete description.

The limited information about the percentage premium method's mechanics, assumptions, and implications in the source record directly contributes to the gaps in the description.

The domain knowledge gaps and mapping rules focused on brevity are secondary factors that exacerbate the impact of the limited source data.

The causal graph illustrates that the lack of contextual details in the source record is the primary driver of the incomplete description. At the same time, the domain knowledge gaps and mapping rules act as compounding factors.

5 Through this root cause analysis, the RAINS agent enables the COGNIGEN-AX system to systematically investigate the underlying reasons for the incomplete description of the "% Premium method ranking" concept. The 5 Whys analysis helps trace the problem back to its origins, the fishbone diagram categorizes the contributing factors, and the causal graph visualizes the relationships between the factors and the observed issue.

Self-Reflection and Critique (Mirror):

10 Relevance and Completeness Assessment "The output effectively captures the ranking concept (90% relevance) but lacks a comprehensive explanation of the percentage premium method (60% completeness)." This assessment highlights that while the output successfully conveys the core concept of the "% Premium method ranking" and its relevance to the insurance provisioning process (as indicated by the high relevance score of 90%), it falls
15 short of providing a complete and detailed explanation of the percentage premium method itself (as evidenced by the low completeness score of 60%).

Improvement Identification: "Improving the description requires additional contextual details on how the method works and its significance in provisioning." Based on the identified limitations, the MIRROR agent recognizes that enhancing the description necessitates
20 including more contextual information. This includes details on the specific calculations and assumptions of the percentage premium method and its implications and importance within the provisioning process.

Potential Solutions: "Alternative data sources or domain knowledge could be leveraged to enhance the description quality." The MIRROR agent suggests exploring alternative data
25 sources or expanding the domain knowledge to address the gaps in the description. This may involve consulting additional insurance provisioning resources, industry guidelines, or subject matter experts to gather the necessary contextual details and insights.

Following the self-analysis report, the MIRROR agent engages in counterfactual reasoning to explore hypothetical scenarios that could improve the output.

30 Scenario 1: Access to Detailed Source Data

The MIRROR agent considers a scenario where the system has access to more comprehensive source data about the percentage premium method.

With detailed information on the method's calculations, assumptions, limitations, and implications readily available in the source record, the output could provide a more thorough and informative description.

- 5 The counterfactual reasoning helps quantify the potential impact of having access to such detailed source data, highlighting how it could significantly enhance the completeness and quality of the output.

Scenario 2: Insurance Provisioning Expertise

The MIRROR agent explores a scenario where the system can access specialized insurance provisioning expertise.

- 10 With a deeper understanding of the intricacies and nuances of the percentage premium method, the system could generate a more accurate, contextually rich, and insightful description.

- 15 The counterfactual reasoning demonstrates how incorporating domain-specific knowledge and expertise could elevate the output's quality, enabling it to provide a more comprehensive and meaningful explanation of the concept.

By engaging in counterfactual reasoning, the MIRROR agent helps the COGNIGEN-AX system understand the potential improvements that could be achieved by addressing the identified limitations. It provides a framework for assessing the impact of different scenarios and guides the system in prioritizing the most effective strategies for enhancing the output.

- 20 In the context of the "% Premium method ranking" example, the counterfactual reasoning emphasizes the value of seeking out more detailed source data and leveraging insurance provisioning expertise to fill the gaps in the description. It highlights how access to such resources could significantly boost the output's completeness and overall quality.

Iterative Refinement:

- 25 Based on the self-reflection insights, the AI reasoner generates an updated output:

- 30 "SARB:ei93:RTG11" | "% Premium method ranking" | "Premium method ranking" | "Ranking of the percentage premium method, which calculates provisions as a percentage of premiums earned, among various methods used for estimating provisions. The ranking reflects the relative importance and reliability of the method in accurately determining provision assumptions." | "Indicates the position and significance of the percentage premium method within the hierarchy of provisioning methods. A higher ranking suggests greater reliance on this method for calculating accurate provisions." | "Used by insurers and rating

agencies to assess the soundness of provisioning assumptions and methodologies. The ranking helps prioritize using different methods based on their effectiveness." | "% Premium method ranking is part of the hierarchy for ranking methods used in provisions assumptions (enumeration) in the Ratings domain. The percentage premium method calculates provisions as a fixed percentage of premiums earned, assuming a stable loss ratio over time." | "High" | "5" | "Very High"

The updated output provides a more comprehensive explanation of the percentage premium method and its implications for provisioning.

Continuous Learning and Adaptation:

- 10 The ALMA agent analyzes the accumulated experiences, identifying patterns such as needing more contextual details in concept descriptions.

It formulates reinforcement plans:

Expand the knowledge base with insurance provisioning methodologies and their comparative advantages.

- 15 Refine the mapping rules to prioritize completeness and context in concept descriptions.

Enhance the self-reflection algorithms to assess description quality accurately and suggest targeted improvements.

The reinforcement plans are implemented, and the AI reasoner's knowledge models and reasoning strategies are adapted based on the insights gained from the "% Premium method ranking" mapping experience.

20

Confidence estimation and calibration

COGNIGEN-AX utilizes advanced techniques to accurately measure and convey uncertainty, enhancing the reliability and quality of AI outputs. This involves incorporating probabilistic modeling techniques to capture the inherent uncertainties and variabilities in AI predictions. The system and methods can generate more precise and well-calibrated confidence estimates by incorporating probability distributions, Bayesian methods, or other approaches for quantifying uncertainty. Furthermore, COGNIGEN-AX includes calibration and validation datasets that can be used to evaluate the accuracy and trustworthiness of confidence estimates. By comparing AI-generated confidence scores with ground truth labels or expert judgments, COGNIGEN-AX can identify and rectify any systematic biases or miscalibrations in the confidence estimation process. Additionally, COGNIGEN-AX offers

25

30

effective methods for communicating confidence levels to users through visual representations, natural language explanations, or interactive interfaces.

FIG. 2 is a flowchart illustrating an example process 200 for generating and applying a confidence score in an artificial intelligence system, according to an embodiment herein.

5 Each step is designed to systematically enhance the confidence with which the AI system operates, reflecting the integration of multiple confidence estimation techniques through a game-theoretical framework.

10 In step 202, the process 200 begins with an input query being processed by a language model to obtain an initial output. This step involves interpreting the input query using natural language understanding techniques to generate a preliminary response or decision output by the AI system.

15 In step 204 multiple confidence estimates are computed for the obtained output, wherein each estimate is derived from distinct computational techniques. This might involve various methodologies such as verbalized confidence assessments, token probability analysis, prompt entropy measurements, semantic output clustering, self-consistency diagnostics, and hidden state divergence metrics.

20 In step 206, multiple confidence estimates obtained from the previous step are aggregated using a game-theoretical approach. This aggregation process utilizes the principles of game theory, potentially applying methods like the Nash Embedding Theorem, to optimize the integration of various confidence measures, assessing their interdependencies to derive a robust, aggregated confidence score.

25 In step 208 the aggregated confidence score is then applied to the artificial intelligence system. This application influences further AI decisions, adjusting operational parameters of the AI system based on the level of confidence determined by the aggregated score. This may involve tuning the AI's response strategies, decision-making thresholds, or other behavioural aspects to enhance reliability and accuracy based on the computed confidence.

30 This method provides a structured approach to enhancing decision-making in AI systems by integrating multiple measures of confidence, thereby addressing the typical challenges of uncertainty and variability in AI-generated outputs. The use of a game-theoretical approach in aggregating confidence scores represents an innovative technique to balance and optimize the contributions of various confidence estimation methods, improving the overall decision quality and reliability of the AI system.

Explainable AI

Explainable AI is a feature of the COGNIGEN-AX system and methods which allows users to comprehend and trust the decisions and predictions made by AI reasoners. To achieve this, the system and methods incorporate interpretable model architectures. These architectures are designed to offer clear and understandable explanations of the reasoning processes.

5 They may include rule-based systems, decision trees, or other transparent model structures that humans can easily follow and interpret. Additionally, the system and methods utilize explanation generation techniques. These techniques translate complex AI reasoning into human-understandable formats like natural language explanations, visual representations, or interactive dialogues. By using these techniques, the system highlights the key factors,
10 evidence, or criteria that influenced the AI's decision-making process. Moreover, COGNIGEN-AX provides user interfaces and visualization tools to facilitate the exploration and understanding of AI explanations. These interfaces enable users to interact with the AI system, ask questions, and receive clarifications on specific aspects of the reasoning process.

15 Continuous learning

COGNIGEN-AX uses dynamic learning algorithms to continuously update AI models as new data or feedback becomes available. These algorithms can handle streaming data, detect changes in concepts, and make real-time adjustments to maintain optimal performance. Additionally, COGNIGEN-AX utilizes transfer learning and domain adaptation techniques,
20 allowing AI reasoners to apply their knowledge and skills to different tasks or domains. By identifying common patterns, representations, or structures, these techniques enable AI systems to reuse and adapt their learned knowledge to new situations, minimizing the need for extensive retraining. Furthermore, COGNIGEN-AX incorporates reinforcement learning and feedback integration mechanisms, enabling AI reasoners to learn from their interactions
25 with the environment and users. Through rewards, penalties, or user feedback, AI systems can refine their decision-making strategies and optimize their behavior based on the outcomes of their actions.

Standardized data interchange

COGNIGEN-AX includes a metadata repository that follows the ISO 11179 standard. This
30 repository allows for the consistent description, management, and exchange of data elements and their associated metadata. It defines clear concepts, names, value domains, and other relevant attributes for data elements. By adhering to the ISO 11179 standard, the system promotes data consistency, reduces ambiguity, and facilitates data sharing and reuse across different AI components and platforms.

In addition to the metadata repository, COGNIGEN-AX offers data mapping and transformation tools. These tools can convert data between different formats, schemas, or ontologies, enabling seamless data integration from various sources and ensuring semantic interoperability among AI reasoners.

- 5 Moreover, COGNIGEN-AX incorporates data quality assessment and validation processes. These processes automatically detect and correct data inconsistencies, errors, or anomalies. They enforce data integrity constraints, perform data cleansing and normalization, and ensure that the data used for AI reasoning meets the required quality standards.

REIMAS:

- 10 REIMAS (Retrospective Experience-driven Iterative Multi-model Agent Self-optimization) is an advanced system and method that generates, evaluates, and iteratively refines AI outputs. REIMAS utilizes an ensemble of specialized agents and methods that collaborate to analyse, assess, and enhance AI-generated content through continuous optimization cycles.

Advantages of REIMAS include:

- 15 Diverse Implementation Modalities: It accommodates various techniques such as prompt-based, JSON-based, quantum-based, and others, making it adaptable across different computational contexts.

Synergistic Agent System and Methods: REIMAS integrates agents like CODA (Concept Extraction, Output, and Drift Analysis), OMFI (Output Monitoring and Feedback Integration),

- 20 MIRROR (Metacognitive Introspection and Reflection for Reasoned Output Refinement), CERES (Contextual Encoding and Reflective Experience Structuring), and ALMA (Adaptive Learning and Model Augmentation), each with unique capabilities, to refine AI outputs.

Cyclic Enhancement Protocol: It rigorously evaluates and iteratively enhances AI outputs by incorporating feedback from multiple agents, aiming to improve quality and consistency.

- 25 Adaptive Optimization: REIMAS facilitates system adaptation by leveraging historical data to derive improvement strategies and fine-tune cognitive systems and methods.

REIMAS agents:

CODA analyzes outputs to ensure conceptual integrity and consistency, as well as reliability. OMFI integrates feedback from different sources to provide actionable improvements,

- 30 enhancing output relevance and user satisfaction. MIRROR uses metacognitive techniques to review decision-making processes, promoting accurate and coherent content generation.

CERES links AI output components to enable better understanding of context, aiding in practical refinement. ALMA coordinates adaptive learning by using experience logs to fine-tune the system and optimize future outputs.

Evaluation & Validation

- 5 The evaluation and validation of each REIMAS component are carefully tailored to align with their specific functionalities and objectives. This involves selecting appropriate ground truth data and evaluation metrics for each component:

CODA's performance is evaluated using ground truth data from established knowledge bases, domain-specific ontologies, and expert annotations. Metrics such as concept
10 extraction accuracy, reasoning validity, and output drift detection precision are used to assess CODA's effectiveness in identifying key concepts, maintaining logical consistency, and detecting anomalies in the generated outputs.

The evaluation of OMFI relies on ground truth data from user studies, expert corpora, and synthetic data that simulate diverse feedback scenarios. Metrics like feedback integration
15 accuracy, quality score correlation, and recommendation relevance are used to measure OMFI's ability to accurately incorporate user feedback, align with expert judgments, and generate meaningful recommendations for output refinement.

MIRROR's performance is assessed using ground truth data derived from expert self-reflections, think-aloud protocols, and simulated metacognitive processes. Metrics such as
20 introspective insight accuracy, suggestion quality, and self-analysis coherence are employed to evaluate MIRROR's effectiveness in generating accurate and coherent introspective insights, as well as providing valuable suggestions for output refinement.

CERES is evaluated using ground truth data from knowledge graphs, linked data repositories, and curated experience encodings. Metrics like semantic link accuracy,
25 provenance traceability, and experience encoding completeness are used to assess CERES's ability to establish accurate semantic connections, maintain traceable provenance, and capture comprehensive experience encodings.

ALMA's performance is evaluated using ground truth data from historical experience logs, simulated reinforcement scenarios, and expert-defined optimization targets. Metrics such as
30 pattern detection accuracy, reinforcement plan efficacy, and model update performance are employed to measure ALMA's effectiveness in identifying meaningful patterns, devising impactful reinforcement strategies, and successfully updating the underlying models.

Integrate multimodal feedback to facilitate self-assessment:

REIMAS incorporates feedback from a variety of sources, including user interactions, system logs, performance metrics, and external evaluations. By considering diverse feedback modalities, the system and methods provide AI reasoners with a comprehensive understanding of their performance and effectiveness. This integration of multi-modal feedback allows AI systems to identify patterns, correlations, and insights that may not be evident from a single feedback source. Additionally, REIMAS employs techniques for weighting and prioritizing different types of feedback based on their relevance, reliability, and significance. This ensures that AI reasoners focus on the most informative and actionable feedback to improve themselves.

Experience-driven self-optimization:

REIMAS incorporates mechanisms that allow AI reasoners to learn from their past experiences and utilize this knowledge to enhance their optimization processes. It includes techniques for capturing and storing relevant experiential data in a structured and accessible format, such as input-output pairs, performance metrics, and user feedback. Furthermore, REIMAS offers algorithms and models to analyse this experiential data and uncover patterns, trends, and improvement opportunities. By applying machine learning and data mining techniques, AI reasoners can identify correlations between specific actions, decisions, or parameters and their impact on performance outcomes. These valuable insights enable the generation of targeted optimization strategies and guide AI systems towards more effective and efficient reasoning processes.

Multi-model adaptive learning:

REIMAS enables AI reasoners to enhance their knowledge and skills continuously. This is achieved by developing algorithms and techniques to effectively integrate and leverage various feedback sources. These sources include explicit user ratings, implicit behavioral signals, system performance metrics, and external evaluations. REIMAS also includes mechanisms to preprocess and normalize this diverse feedback data, ensuring it can be meaningfully combined and analysed. Additionally, REIMAS provides adaptive learning models that can dynamically adjust their parameters, weights, or structures based on the integrated feedback. These models prioritize the most informative and reliable feedback signals for learning. Furthermore, REIMAS incorporates techniques to detect and mitigate concept drift, allowing AI reasoners to adapt to changing data distributions, user preferences, or task requirements over time.

Collaborative human-AI feedback assimilation:

REIMAS improves the effectiveness of AI reasoners by incorporating human expertise and guidance into their self-improvement process. This is achieved by developing interfaces and protocols that enable smooth and intuitive interactions between human users and AI systems. The system and methods allow users to provide qualitative feedback, suggestions, and critiques on the performance, reasoning processes, or outputs of the AI reasoners. Additionally, REIMAS includes techniques for processing and interpreting human feedback, extracting relevant insights, and aligning them with the AI reasoners' internal representation and optimization objectives. Furthermore, REIMAS incorporates methods for establishing trust and transparency in the collaboration between humans and AI, ensuring that users can understand and validate the self-improvement decisions and outcomes of the AI reasoners. This may include generating explanations, visualizations, or interactive dialogues that clarify the rationale behind specific optimizations or adaptations.

CODA (Concept Extraction, Output, and Drift Analysis) Methods:

CODA employs advanced techniques for semantic analysis, reasoning, output drift detection, confidence estimation, and interactive concept learning to extract key concepts from prompts.

Concept Extraction:

CODA meticulously extracts essential concepts from prompts, including ideas, inputs, outputs, reasoning chains, descriptions, activities, and personas. It captures a comprehensive context that can adapt to various input formats (prompt-based, JSON-based, quantum-based), enabling seamless system integration. The extracted concepts are represented using the ISO 11179 metadata structure to ensure standardization and interoperability. To extract key concepts from prompts, CODA employs advanced semantic parsing techniques such as dependency parsing with the spaCy library and named entity recognition using the BERT model. The extraction process follows a multi-step approach:

Tokenization: The prompt is tokenized into individual words using the NLTK library.

Part-of-speech tagging: Each token is assigned a part-of-speech tag using the Stanford POS tagger.

Dependency parsing: The dependency tree is constructed using the spaCy library, identifying the relationships between tokens.

Named entity recognition: Named entities are extracted using the BERT model fine-tuned on the CoNLL-2003 dataset.

Concept identification: Relevant concepts are identified based on predefined rules and heuristics, considering factors such as part-of-speech tags, dependency relations, and named entity types.

The concept extraction process has a time complexity of $O(n)$, where n is the number of tokens in the prompt. The space complexity is also $O(n)$, as the dependency tree and named entity information are stored for each token.

Semantic Analysis and Reasoning:

CODA utilizes embeddings, ontology alignment, and knowledge graphs to measure semantic similarity between concepts. CODA employs rule-based systems, inference engines, and probabilistic approaches to perform reasoning tasks, uncover relationships, infer knowledge, and validate output consistency. CODA determines the meaning and usage of each extracted concept based on its description, exploring contexts, domains, applications, synonyms, antonyms, and semantic relationships. This comprehensive understanding enables accurate similarity measurements and reasoning. The resulting meanings and usages are represented using the ISO 11179 metadata structure.

The semantic analysis and reasoning process in CODA combines word embeddings and knowledge graphs. The following steps are involved:

1. Word embedding generation: Each extracted concept is represented as a dense vector using pre-trained word embeddings like Word2Vec or GloVe.

2. Semantic similarity calculation: The cosine similarity between concept embeddings is computed to measure their semantic relatedness. The similarity score between concepts c_i and c_j is given by: $\text{sim}(c_i, c_j) = (c_i \cdot c_j) / (\|c_i\| \times \|c_j\|)$

3. Knowledge graph integration: Relevant domain knowledge is incorporated by mapping the extracted concepts to entities in a knowledge graph, such as ConceptNet or WordNet. The knowledge graph is traversed to infer relationships and reasoning paths between concepts.

4. Reasoning and inference: Logical reasoning rules and inference mechanisms, such as first-order logic and description logic, are applied to derive new knowledge and conclusions based on the semantic relationships and knowledge graph structure.

The worst-case time complexity of the semantic analysis and reasoning process is $O(n^2)$, where n is the number of extracted concepts, due to the pairwise similarity calculations. The space complexity is $O(n)$ for storing the concept embeddings and knowledge graph information.

- 5 For semantic similarity, CODA uses cosine similarity between concept embeddings:

$$\text{sim}(c_i, c_j) = (c_i \cdot c_j) / (\|c_i\| \times \|c_j\|)$$

where c_i and c_j are the embedding vectors for concepts i and j .

For reasoning, CODA employs first-order logic inference rules of the form:

$$\forall x (P(x) \wedge Q(x) \Rightarrow R(x))$$

- 10 where P , Q , and R are predicates representing concept properties or relationships.

Output Drift Assessment:

- CODA detects and mitigates output drift risks, such as hallucinations, by categorizing drift into comprehensive categories. It uses anomaly detection, statistical analysis, and machine learning models to identify inconsistencies and deviations, allowing for corrective actions to produce accurate and reliable outputs. By assessing each concept and activity against predefined output drift categories (e.g., numerical errors, acronym confusion, fictitious references, unverified claims, location inaccuracies, and timeline contradictions), CODA generates both genuine and hallucinated responses. It then calculates the divergence between these responses using distance metrics like Jensen-Shannon divergence (JSD), which is given by $\text{JSD}(P \parallel Q) = (1/2) \times (\text{D_KL}(P \parallel M) + \text{D_KL}(Q \parallel M))$. Here, P and Q represent the probability distributions of the genuine and hallucinated responses, $M = (1/2) \times (P + Q)$, and D_KL is the Kullback-Leibler divergence.

- To provide an output drift assessment rationale, CODA utilizes the ISO 11179 metadata structure. If the JSD indicates a high divergence (e.g., $\text{JSD} > 0.5$), CODA determines the required level of human participation (very high, high, medium, low), specific human activities needed, documents or resources necessary for human assistance, and the persona best suited to provide human aid. This human assistance recommendation also follows the ISO 11179 metadata structure to facilitate collaboration.

- CODA employs statistical analysis and anomaly detection techniques to assess output drift and identify inconsistencies. The main steps involved are:

Baseline establishment: A representative sample of historical outputs is used to establish a baseline distribution of output characteristics, such as topic distributions or linguistic features.

Drift detection: Statistical tests, such as the Kullback-Leibler divergence or the Kolmogorov-Smirnov test, may be applied to compare the current output distribution against the baseline. The Kullback-Leibler divergence between distributions P and Q is given by $KL(P \parallel Q) = \sum_i P(i) * \log(P(i) / Q(i))$.

Anomaly detection: Anomaly detection algorithms, such as Isolation Forest or Local Outlier Factor, are used to identify outliers or anomalous outputs that deviate significantly from the expected patterns.

Drift assessment: The severity and impact of the detected drift are assessed based on predefined thresholds and criteria, considering factors such as the magnitude of the divergence and the frequency of anomalies.

The time complexity of the output drift assessment process depends on the specific algorithms used. For example, the Isolation Forest algorithm has an average time complexity of $O(n \log n)$ for training and $O(\log n)$ for prediction, where n is the number of output samples. The space complexity is $O(n)$ for storing the output representations and anomaly detection models.

Confidence Estimation:

CODA uses Bayesian inference, Monte Carlo simulations, and uncertainty quantification to estimate confidence intervals and probabilistic bounds for extracted concepts and outputs. This helps in assessing the trustworthiness of the results and identifying high-confidence outputs as well as areas that require further investigation. CODA determines the confidence level for each concept based on factors such as the quality of the concept description, the relevant supporting sentences from the input data, the assessment of output drift, and the expertise level of the associated persona. The confidence of the estimated concept is represented using the ISO 11179 metadata structure.

To estimate confidence, CODA employs a Bayesian approach by updating the prior probability $P(H)$ of a concept's correctness to a posterior probability $P(H|E)$ based on new evidence E . The formula used is $P(H|E) = (P(E|H) \times P(H)) / P(E)$, where $P(E|H)$ is the likelihood of observing the evidence given the concept's correctness, and $P(E)$ is the marginal probability of the evidence.

To estimate confidence, CODA combines probabilistic modeling and uncertainty quantification techniques. The key steps involved are as follows:

1. Probability calibration: The raw probabilities or scores associated with extracted concepts or generated outputs are calibrated using techniques like Platt scaling or isotonic regression to obtain well-calibrated confidence estimates.

2. Uncertainty quantification: Bayesian inference techniques such as Markov Chain Monte Carlo (MCMC) or variational inference are used to estimate the uncertainty related to the predictions. The uncertainty is quantified by computing the posterior distribution over the model parameters based on the observed data.

3. Confidence score calculation: The calibrated probabilities and uncertainty estimates are combined to calculate an overall confidence score for each concept or output. The confidence score can be computed as the expected value of the posterior distribution or using other aggregation methods such as the harmonic mean.

The time complexity of the confidence estimation process depends on the specific techniques used. For instance, MCMC methods have a time complexity of $O(n)$, where n is the number of sampling iterations. The space complexity is $O(m)$, where m is the number of model parameters, as the posterior distributions need to be stored.

The COGNIGEN-AX system implements several techniques for estimating confidence in AI model outputs and calibrating the confidence scores.

Token Probabilities (TPA):

Language models generate outputs token-by-token. The TPA technique extracts the softmax probabilities of each generated token from the model's output distribution.

The overall confidence score is calculated by taking the product of the token probabilities for the entire output sequence (or the sum of log probabilities to avoid underflow).

Calibration is done through temperature scaling, where the logits are divided by a learnable temperature parameter before applying softmax to align the probabilities with true correctness likelihoods.

Example Method of Estimating Confidence:

The language model generates output token-by-token in an autoregressive manner.

For each token, the model produces a probability distribution over the vocabulary.

The TPA technique extracts the softmax probability $p(y_t | y_{<t}, x)$ for the generated token y_t given the previous tokens $y_{<t}$ and input x

The overall sequence probability is calculated as the product: $P(y|x) = \prod_t p(y_t | y_{<t}, x)$

To avoid numerical underflow issues, log probabilities are typically used: $\log P(y|x) = \sum_t \log p(y_t | y_{<t}, x)$

Higher sequence probabilities indicate more confidence in the generated output.

Calibration via temperature scaling involves learning a temperature parameter T during training to divide logits before applying softmax: $p(y_t | y_{<t}, x) = \text{softmax}(h_t / T)$

Prompt Entropy (PE):

- 10 PE prompts the language model with different label options for the input and calculates the entropy of the predicted probability distribution over those options.

Higher entropy indicates the model is more uncertain about the correct label, while low entropy suggests high confidence.

- 15 The entropy values are mapped to confidence scores with a predefined function or learned mapping method.

Calibration is achieved by optimizing a temperature parameter on a validation set to minimize the calibration loss between entropies and actual probabilities.

Method:

- 20 The input prompt is augmented with enumerated label options from the target taxonomy/classes.

The PE technique applies masked language modeling (MLM) to predict a probability distribution $p(y|x)$ over these label options given the prompt x

The entropy $H(p)$ is calculated as: $H(p) = -\sum_y p(y|x) \log p(y|x)$

- 25 A higher entropy value indicates more uncertainty, as probabilities are spread across many labels

Lower entropy suggests the model is confident in assigning high probability to one label

To calibrate, a temperature parameter T is optimized on a validation set to minimize: $\sum_i -p_i \cdot \log(p_i^{1/T})$

Verbalized Confidence (VCA):

The VCA technique uses pattern matching to look for explicit textual confidence statements like "I am 80% confident that..." in the model's output.

It assigns numerical scores to the verbalized statements based on a predefined mapping.

- 5 VCA incorporates the Hidden State Divergence Metric (HSDM) to improve reliability. This metric measures the divergence between hidden state representations of genuine and adversarial/hallucinated responses.

Semantic Output Clustering (SOC):

SOC generates diverse output sequences using sampling techniques.

- 10 It computes semantic embeddings for these outputs and applies clustering algorithms like K-means.

Cluster tightness/entropy measures the consistency/certainty of the outputs.

Well-separated, low-entropy clusters suggest higher confidence in the predictions.

Method:

- 15 SOC generates N diverse output sequences $\{y_1, y_2, \dots, y_N\}$ via sampling techniques like top-k, nucleus sampling

It computes vector representations/embeddings $\{e_1, e_2, \dots, e_N\}$ for these outputs

A clustering algorithm (e.g., K-means) is applied to cluster the embeddings into K clusters.

The clustering quality/tightness reflects consistency/certainty of predictions.

- 20 Metrics like cluster entropy and silhouette score are used to quantify the confidence.

Self-consistency measures prediction consistency across rephrased prompts using Fleiss' Kappa.

Self-Consistency (SC):

- 25 SC evaluates the consistency of predictions when the input is slightly varied, e.g., by paraphrasing.

It computes agreement metrics like Fleiss' Kappa across multiple input variations.

High agreement indicates the model is making consistent, confident predictions despite superficial input changes.

Example:

Token Probabilities (TPA):

- 5 Step 1: Generate the output token by token and extract the softmax probability for each token.

$$p(\text{"Ranking"}) = 0.85$$

$$p(\text{"of"} \mid \text{"Ranking"}) = 0.95$$

$$p(\text{"the"} \mid \text{"Ranking of"}) = 0.98$$

- 10 ... $p(\text{"High"} \mid \text{"...assumptions (enumeration) in the Ratings domain."} \mid \text{"Medium"} \mid \text{"4"} \mid) = 0.9$

Step 2: Calculate the overall sequence probability as the product of token probabilities.

$$P(\text{"Ranking of the percentage premium method..."}) = 0.85 * 0.95 * 0.98 * \dots * 0.9 = 0.65$$

Step 3: Use log probabilities to avoid numerical underflow.

$$\log P(\text{"Ranking of the percentage premium method..."}) = \log(0.85) + \log(0.95) + \log(0.98)$$

- 15 + ... + $\log(0.9) = -0.43$

Step 4: Calibrate the token probabilities using temperature scaling.

Assuming $T = 1.1$, the calibrated token probabilities are:

$$p(\text{"Ranking"}) = \text{softmax}(\text{logit}(\text{"Ranking"}) / 1.1)$$

$$p(\text{"of"} \mid \text{"Ranking"}) = \text{softmax}(\text{logit}(\text{"of"}) / 1.1)$$

- 20 ... $p(\text{"High"} \mid \text{"...assumptions (enumeration) in the Ratings domain."} \mid \text{"Medium"} \mid \text{"4"} \mid) = \text{softmax}(\text{logit}(\text{"High"}) / 1.1)$

Prompt Entropy (PE):

Step 1: Augment the input prompt with label options from the target taxonomy.

Augmented prompt: "The % Premium method ranking is [MASK]. Options: High, Medium,

- 25 Low"

Step 2: Apply masked language modeling (MLM) to predict a probability distribution over the label options.

Predicted distribution: $p(y|x) = [0.7, 0.25, 0.05]$

Step 3: Calculate the entropy of the predicted distribution.

5 Entropy: $H(p) = -(0.7 * \log(0.7) + 0.25 * \log(0.25) + 0.05 * \log(0.05)) = 0.80$

Step 4: Interpret the entropy value.

An entropy of 0.80 indicates relatively low uncertainty, as the probabilities are concentrated on the "High" option.

10 Step 5: Calibrate the entropy values by optimizing a temperature parameter T on a validation set.

Assuming $T = 0.9$, the calibrated probabilities are:

$$p(\text{"High"}|x)^{1/0.9} = 0.7^{1/0.9} = 0.74$$

$$p(\text{"Medium"}|x)^{1/0.9} = 0.25^{1/0.9} = 0.32$$

$$p(\text{"Low"}|x)^{1/0.9} = 0.05^{1/0.9} = 0.15$$

15 Verbalized Confidence (VCA):

Step 1: Look for explicit textual confidence statements in the model's output using pattern matching.

Matched statement: "High"

20 Step 2: Assign a numerical score to the verbalized statement based on a predefined mapping.

Mapping: "High" -> 0.8

Step 3: Calculate the Hidden State Divergence Metric (HSDM) between genuine and adversarial/hallucinated responses.

25 Assume the HSDM between the genuine and adversarial responses: "The % Premium method ranking is Low." is 0.4.

Step 4: Adjust the verbalized confidence score based on the HSDM value.

Adjusted confidence score: $0.8 * (1 - 0.4) = 0.48$

Semantic Output Clustering (SOC):

Step 1: Generate N diverse output sequences using sampling techniques.

In this example, N = 3 output sequences are generated:

"The % Premium method ranking is High."

5 "The percentage premium method is ranked as a critical method for calculating provisions assumptions."

"Among the various methods used for estimating provisions, the % Premium method is considered highly significant."

Step 2: Compute vector representations/embeddings for the generated outputs.

10 Embeddings:

$e_1 = [0.1, 0.3, \dots, 0.2]$

$e_2 = [0.2, 0.4, \dots, 0.1]$

$e_3 = [0.1, 0.2, \dots, 0.3]$

15 Step 3: Apply a clustering algorithm (e.g., K-means) to cluster the embeddings into K clusters.

Assume $K = 2$, and the outputs are clustered as:

Cluster 1: ["The % Premium method ranking is High.", "Among the various methods used for estimating provisions, the % Premium method is considered to be of high significance."]

20 Cluster 2: ["The percentage premium method is ranked as a highly important method for calculating provisions assumptions."]

Step 4: Evaluate the clustering quality/tightness using metrics like cluster entropy or silhouette score.

Assume the cluster entropy is 0.3, indicating relatively high consistency within clusters.

Step 5: Interpret the clustering metrics to determine the confidence in the predictions.

25 A cluster entropy of 0.3 suggests moderate to high confidence, as each cluster's outputs are similar.

Self-Consistency (SC):

Step 1: Generate predictions for slightly varied input prompts.

Input variations:

"What is the ranking of the % Premium method among various methods used for calculating provisions assumptions?"

- 5 "How important is the percentage premium method in estimating provisions compared to other methods?"

"The significance of the % Premium method within the hierarchy of provisioning methods is [MASK]."

Generated outputs:

- 10 "The % Premium method ranking is High."

"The percentage premium method is considered highly important for calculating provisions assumptions."

"High"

Step 2: Calculate agreement metrics like Fleiss' Kappa across the generated outputs.

- 15 Assume the observed agreement P_a is 0.9 and the expected agreement P_e is 0.4.

$$\kappa = (0.9 - 0.4) / (1 - 0.4) = 0.83$$

Step 3: Interpret the agreement metric to determine the consistency and confidence of the predictions. A Fleiss' Kappa value of 0.83 indicates substantial agreement and consistency across the outputs, suggesting high confidence in the predictions.

- 20 Interactive Concept Learning:

CODA facilitates interactive and adaptive concept learning between AI and humans through intuitive interfaces and collaborative workflows. CODA employs active learning strategies, seeking human input on ambiguous concepts to continuously refine the knowledge base and align it with human understanding. CODA selects the most informative concepts for human

25 feedback using uncertainty sampling, querying the concepts with the highest uncertainty in their predicted labels: $x^* = \operatorname{argmax}_x (1 - P(\hat{y}|x))$, where x^* represents the most informative concept, $\hat{y}$ is the predicted label, and $P(\hat{y}|x)$ is the probability of the predicted label given the idea.

CODA supports interactive concept learning by actively seeking user feedback and adapting its knowledge base. The main steps involved are:

Active learning: CODA identifies informative or uncertain concepts that would benefit from user feedback using active learning strategies, such as uncertainty sampling or query-by-committee. The most informative concept is selected based on the expected information gained or the disagreement among multiple models.

User feedback: CODA prompts the user to provide input on the selected concepts, such as confirming their relevance, providing additional examples, or specifying relationships with other ideas.

Knowledge base updating: The user feedback is incorporated into CODA's knowledge base by updating the concept representations, modifying semantic relationships, or adding new concepts. The knowledge base is dynamically expanded and refined using the iterative feedback process.

Concept relearning: CODA retrains or fine-tunes its models using the updated knowledge base to improve its understanding and generation capabilities of concepts. The relearning process adapts the models to the new information and feedback provided by users.

The time complexity of the interactive concept learning process depends on the size of the knowledge base and the frequency of user interactions. The space complexity of the knowledge base is determined by the number of stored user feedback instances.

CODA Business Layer:

Actors

The key actors involved in the CODA component of the REIMAS system and methods are the Domain Expert, Data Scientist, and AI Engineer. These actors contribute their expertise and play important roles in the concept extraction, semantic analysis, output drift assessment, and confidence estimation processes.

Services

CODA offers the following key services:

1. Concept Extraction: It automatically extracts relevant concepts from textual prompts, making it easier to process and understand user inputs.

2. Semantic Analysis: It provides semantic reasoning and inference services to identify relationships between concepts, derive new knowledge, and support decision-making.

3. Output Drift Assessment: It helps detect and mitigate output drift to ensure the consistency and reliability of the generated outputs over time.

- 5 4. Confidence Estimation: It provides confidence scores and uncertainty estimates for the extracted concepts and generated outputs, enabling informed decision-making and trust in the system's outputs.

Processes

- 10 The CODA component facilitates two processes: Prompt Analysis and Concept Extraction, and AI Output Generation and Evaluation. These processes involve analyzing task prompts, extracting relevant concepts, generating AI outputs, and evaluating those outputs for accuracy, consistency, and reliability. The processes facilitated by CODA are:

- 15 Data Ingestion and Preprocessing: This process involves collecting and preprocessing textual prompts, applying tokenization, part-of-speech tagging, and named entity recognition to prepare the data for concept extraction.

Model Training and Deployment: This process focuses on training and deploying machine learning models for concept extraction, semantic analysis, and output drift assessment. It involves data preparation, model selection, hyperparameter tuning, and performance evaluation.

- 20 Output Monitoring and Feedback: This process involves continuously monitoring the generated outputs, assessing their quality and consistency, and collecting user feedback for improvement. It enables the iterative refinement of the models and knowledge base based on real-world usage and domain expertise.

Functions

- 25 CODA supports several business functions, including Data Ingestion and Pre-processing, AI Model Training and Deployment, and Output Monitoring and Evaluation. These functions ensure that input data is handled appropriately, AI models are trained and deployed efficiently, and the generated outputs are continuously monitored and evaluated for quality and performance. The business functions supported by CODA are:

1. Enhanced User Experience: By extracting relevant concepts and providing semantic reasoning, CODA enables more natural and intuitive user interactions, improving the overall user experience.

5 2. Improved Decision Support: CODA's semantic analysis and reasoning capabilities help users make informed decisions by providing insights, recommendations, and knowledge inference based on the extracted concepts.

3. Quality Assurance: The functionalities of output drift assessment and confidence estimation ensure the quality, consistency, and reliability of the generated outputs, reducing the risk of errors or inconsistencies.

10 4. Continuous Improvement: The interactive concept learning process allows CODA to continuously learn and adapt based on user feedback, improving its performance and domain understanding over time.

Objects

The main business objects involved in CODA are:

15 1. Task Prompt: It represents the textual input provided by the user, containing the instructions or queries to be processed by the system.

2. Extracted Concepts: These are the key concepts extracted from the task prompt, serving as the foundation for semantic analysis and reasoning.

20 3. Knowledge Base: It is a structured repository of domain knowledge, including concepts, relationships, and rules, used for semantic reasoning and inference.

4. Generated Outputs: These are the final outputs produced by the system based on the extracted concepts and semantic analysis, such as answers, recommendations, or actions.

CODA Application Layer:

Components of CODA

25 The Concept Extractor component within CODA extracts concepts from textual prompts using natural language processing techniques and machine learning models. The extraction process is guided by defined rules and heuristics.

The Concept Extractor component interacts with the Semantic Analyzer component, which performs semantic analysis and reasoning on the extracted concepts. By utilizing knowledge

graphs, ontologies, and inference engines, it identifies relationships, derives new knowledge, and supports decision-making processes.

The Output Drift Detector component is responsible for assessing output drift and detecting inconsistencies in the generated outputs. It compares the current outputs to historical

5 patterns using statistical analysis and anomaly detection algorithms, enabling the identification of significant deviations.

The Confidence Estimator component estimates confidence scores and uncertainty measures for the extracted concepts and generated outputs. It achieves this by employing probabilistic models, calibration techniques, and Bayesian inference. The confidence

10 estimates provided are well-calibrated and interpretable.

Services

CODA offers a range of application services to enhance functionality and facilitate integration with other components or systems. These key application services are as follows:

1. Concept Extraction Service: This service allows for the extraction of concepts from textual prompts. It provides an interface that enables other components or external systems to

15 utilize the concept extraction functionality.

2. Semantic Reasoning Service: With this service, users can explore relationships between concepts and derive new knowledge. It offers semantic reasoning and inference capabilities based on defined ontologies and rules.

3. Output Drift Assessment Service: This service provides methods for assessing output drift and detecting inconsistencies. It allows for the monitoring and evaluation of the quality and reliability of generated outputs.

20

4. Confidence Estimation Service: Users can utilize this service to estimate confidence scores and uncertainty measures. It enables the integration of confidence information into decision-making processes and user interfaces.

25

Interfaces

CODA provides several application interfaces, which include the Task Prompt Interface for receiving input data, the Domain Knowledge Interface for accessing relevant domain knowledge, and the Reporting and Visualization Interface for presenting analysis results and insights. CODA exposes the following application interfaces:

30

Concept Extractor Interface: This interface defines the methods for extracting concepts from textual prompts. It specifies the input and output formats, as well as any required parameters.

5 Semantic Analyzer Interface: This interface describes the methods for performing semantic analysis and reasoning. It includes the input concepts, knowledge base, and desired inference types.

Output Drift Detector Interface: This interface specifies the methods for assessing output drift. It includes the input output data, comparison metrics, and drift detection thresholds.

10 Confidence Estimator Interface: This interface defines the methods for estimating confidence scores and uncertainty measures. It specifies the input data, confidence estimation techniques, and output formats.

Data:

15 CODA utilizes various types of application data, including Concept Metadata, Output Metadata, and Feedback Metadata. These data structures store information related to extracted concepts, generated outputs, and received feedback or evaluations. The application data models used in CODA are:

Prompt: This represents the user's textual input, which includes instructions or queries to be processed.

20 Concept: This represents an extracted concept and includes its textual representation, semantic type, and associated metadata.

Semantic Graph: This represents the semantic relationships between concepts, capturing the knowledge base and ontology structure.

Output: This represents the generated output, including the textual content, associated concepts, and confidence scores.

25 Functions:

CODA incorporates the following key application functions:

Concept Extraction: This function extracts concepts such as inputs, outputs, reasoning, descriptions, activities, and personas from task prompts. It creates concept extraction methods, outlines the extraction process, and prioritizes and lists the extracted concepts.

Semantic Analysis: This function derives the meanings and usages of concepts based on descriptions. It measures semantic similarity between concepts using embeddings, ontologies, and knowledge graphs. It also performs reasoning tasks to uncover relationships, infer knowledge, and validate consistency.

- 5 Output Drift Assessment: This function assesses concepts and activities against different categories of output drift, such as numerical errors, fictitious references, and inconsistencies. It generates genuine and hallucinated responses for drift assessment, calculates the divergence between genuine and hallucinated responses, and provides ratings and rationales for output drift assessment.
- 10 Confidence Estimation: This function estimates confidence intervals and probabilistic bounds for concepts and outputs. It applies techniques like Bayesian inference, Monte Carlo simulations, and uncertainty quantification to determine the confidence levels.

- Interactive Concept Learning: This function enables interactive and adaptive concept learning between AI and humans. It uses active learning strategies to gather input on ambiguous concepts. It recommends human assistance based on output drift and confidence levels. It also identifies support sentences from input data to explain or contextualize concepts.
- 15

The CODA application functions may be implemented via:

- extractConcepts(prompt: Prompt): List[Concept]: This function takes a textual prompt as input and returns a list of extracted concepts. The extraction is based on predefined rules and heuristics.
- 20

- performSemanticAnalysis(concepts: List[Concept], knowledgeBase: SemanticGraph): SemanticGraph: This function performs semantic analysis and reasoning on the extracted concepts. It utilizes the knowledge base and ontologies to identify relationships and derive new knowledge.
- 25

detectOutputDrift(outputs: List[Output], historicalData: List[Output]): DriftScore: This function assesses the output drift by comparing the current outputs with historical patterns. It returns a drift score indicating the level of inconsistency.

- estimateConfidence(output: Output, modelParameters: ConfidenceParameters): ConfidenceScore: This function estimates the confidence score and uncertainty measures for a given output. It uses specified confidence estimation techniques and model parameters.
- 30

CODA's modular and service-oriented design allows for flexibility, scalability, and integration with other components and systems. The well-defined interfaces and data models facilitate seamless communication and data exchange. The separation of concerns enables independent development and maintenance of each component.

5 OMFI (Output Monitoring and Feedback Integration) Methods

OMFI combines feedback signals from various sources, including user interactions, performance metrics, and external evaluations, to generate actionable recommendations for improving AI-generated outputs. These recommendations guide other agents in making targeted enhancements to enhance the quality, relevance, and user satisfaction.

10 Feedback Integration:

OMFI collects feedback from automated tests, user surveys, and expert reviews. It processes and consolidates this feedback into a structured format for analysis. The feedback metadata is represented using the ISO 11179 standard to ensure consistency and interoperability.

15 OMFI utilizes a weighted average approach to integrate feedback scores from different sources:

$$F_i = (\sum_j w_j \times f_{ij}) / (\sum_j w_j)$$

where F_i represents the integrated feedback score for output i , w_j is the weight assigned to feedback source j , and f_{ij} is the feedback score from source j for output i .

20 Quality Assessment:

OMFI analyzes the consolidated feedback and evaluates the quality of the AI-generated outputs based on predefined criteria such as accuracy, relevance, completeness, consistency, and appropriateness. It assigns quality scores or ratings to the outputs using a multi-criteria decision analysis (MCDA) approach, such as the Analytic Hierarchy Process (AHP).

The AHP method involves the following steps:

1. Create a hierarchy of quality criteria and sub-criteria.
2. Conduct pairwise criteria comparisons to determine their relative importance, using a scale from 1 to 9.

30 3. Calculate the eigenvector of the pairwise comparison matrix to obtain criteria weights.

4. Evaluate each output alternative against the criteria, assigning scores on a predefined scale.

5. Calculate the weighted sum of criteria scores for each alternative to obtain overall quality scores:

5 $Q_i = \sum_j w_j \times s_{ij}$

where Q_i represents the quality score for output i , w_j is the weight of criterion j , and s_{ij} is the score of output i against criterion j .

Recommendation Generation:

Based on the quality assessments and feedback analysis, OMFI generates actionable
10 recommendations for improving the output quality. It identifies specific areas or aspects that require enhancement, prioritizes the recommendations based on their potential impact and feasibility, and provides guidance for implementing the suggested improvements.

OMFI employs a rule-based expert system to generate recommendations, using IF-THEN
15 rules that map quality assessment results and feedback patterns to specific improvement actions. The rules are structured as follows:

IF (condition_1) AND (condition_2) AND ... AND (condition_n) THEN (recommendation)

For example:

IF (accuracy_score < 0.8) AND (relevance_score < 0.7) THEN (recommendation: "Improve
20 the accuracy and relevance of the output by incorporating more domain-specific knowledge and contextual information.")

Output Classification:

OMFI classifies the AI-generated outputs (e.g., excellent, good, needs improvement, poor, unacceptable) based on their quality scores and predefined thresholds. This classification helps determine the appropriate course of action for each output. The classification uses a
25 decision tree algorithm, where each internal node represents a quality score threshold, and each leaf node represents an output category. The decision tree is traversed from the root to a leaf, following the branches corresponding to the output's quality score until a category is reached.

OMFI Design: Business Layer

30 Actors:

The key business actors involved in the OMFI component include Domain Experts, Business Analysts, and End-Users. Domain Experts provide subject matter expertise for evaluating the outputs, Business Analysts analyze the feedback and performance metrics, and End-Users interact with the system and provide user feedback.

5 Services:

OMFI offers business services such as output evaluation and feedback integration. This service enables the comprehensive assessment of AI-generated outputs by consolidating feedback from various sources, such as automated tests, user surveys, and expert reviews, and integrating this feedback to enhance output quality and user satisfaction. OMFI offers

10 the following business services:

Output Evaluation and Feedback Integration: This service enables the comprehensive assessment of AI-generated outputs by consolidating feedback from various sources, such as automated tests, user surveys, and expert reviews. The integrated feedback provides a holistic view of the output quality and user satisfaction.

15 Quality Assurance and Continuous Improvement: OMFI ensures the quality of AI-generated outputs by evaluating them against predefined criteria and industry standards. It also facilitates continuous improvement by incorporating feedback and recommendations into the iterative output generation process.

Processes:

20 The primary business processes facilitated by OMFI are output monitoring and iterative refinement. This process involves monitoring the AI-generated outputs, gathering feedback from multiple sources, analyzing the feedback, and using the insights to refine and improve the outputs iteratively. The primary business processes facilitated by OMFI are:

25 Output Monitoring and Evaluation: This process involves monitoring the AI-generated outputs, gathering feedback from multiple sources, analyzing the feedback, and assessing the output quality based on predefined criteria.

Iterative Output Refinement: OMFI supports the iterative refinement of AI-generated outputs by incorporating the insights and recommendations derived from the feedback analysis. This process ensures that the outputs are continuously improved to meet business objectives and
30 user expectations.

Functions:

OMFI supports the business functions of Output Quality Assurance and Continuous Improvement. It ensures the quality of AI-generated outputs by evaluating them against predefined criteria and industry standards. It also enables continuous improvement by incorporating feedback and recommendations into subsequent iterations of the output generation process. OMFI supports the following business functions:

Quality Control: It ensures that the AI-generated outputs meet the required quality standards by evaluating them against established criteria and metrics.

User Satisfaction Management: OMFI helps to manage and improve user satisfaction by actively seeking and incorporating user feedback into the output generation process.

- 10 Performance Optimization: By identifying areas for improvement and providing actionable recommendations, OMFI contributes to the optimization of the AI system's performance, helping to achieve better business results.

Objects:

- 15 The key business objects involved in the OMFI component include AI Outputs, Feedback Metadata, Quality Assessments, and Improvement Recommendations. AI Outputs are generated results or responses; Feedback Metadata captures the sources and details of the feedback received; Quality Assessments evaluate output quality; and Improvement Recommendations provide actionable suggestions for enhancing the outputs. The key business objects involved in the OMFI component include:

- 20 AI Outputs: These are the generated results or responses produced by the AI system, which are subject to evaluation and refinement.

Feedback Metadata: This object captures the details and metadata associated with the feedback received from various sources, such as feedback type, source, timestamp, and specific comments or ratings.

- 25 Quality Assessments: These are the evaluations of the AI-generated outputs based on predefined criteria and metrics, indicating the level of quality and areas for improvement.

Improvement Recommendations: These are the actionable suggestions and guidelines provided by OMFI to enhance the quality and performance of the AI-generated outputs.

OMFI: Application Layer

The core OMFI (Output Monitoring and Feedback Integration) component, which application component encapsulates a functionality related to feedback integration, quality assessment, recommendation generation, and output classification.

Services:

- 5 OMFI provides the following application services: Feedback Integration Service, Quality Assessment Service, and Recommendation Generation Service. These services facilitate the integration of feedback from various sources, the assessment of output quality, and the generation of recommendations for improvement. OMFI provides the following application services:
- 10 Feedback Integration Service: This service is responsible for collecting and consolidating feedback from various sources, such as automated tests, user surveys, and expert reviews. It preprocesses and normalizes the feedback data to ensure consistency and compatibility.
- Quality Assessment Service: This service evaluates the quality of the AI-generated outputs based on predefined criteria and metrics. It applies the Analytic Hierarchy Process (AHP) to
- 15 calculate the overall quality scores considering the relative importance of different criteria.
- Recommendation Generation Service: This service generates actionable recommendations for improving the output quality. It utilizes a rule-based expert system to map specific quality assessment results and feedback patterns to corresponding improvement actions.
- Output Classification Service: This service classifies the AI-generated outputs into
- 20 predefined categories based on their quality scores and predefined thresholds. It employs a decision tree algorithm to determine the appropriate category for each output.

Interfaces:

- OMFI exposes several application interfaces, including the User Feedback Interface for collecting feedback from end-users, the Expert Review Interface for soliciting input from
- 25 domain experts, and the Reporting and Visualization Interface for presenting quality assessments and recommendations. OMFI exposes the following application interfaces:

Feedback Integration Interface: This interface allows external components or systems to provide feedback data to OMFI. It defines the format and structure of the feedback data and the methods for submitting and retrieving feedback.

Quality Assessment Interface: This interface enables external components or systems to request quality assessments of AI-generated outputs. It specifies the input format for the outputs and the output format for the quality assessment results.

- 5 Recommendation Interface: This interface allows external components or systems to retrieve the generated recommendations for improving the output quality. It defines the format and structure of the recommendation data.

Classification Interface: This interface enables external components or systems to request the classification of AI-generated outputs into predefined categories. It specifies the input format for the outputs and the output format for the classification results.

10 Data:

- The application data used within the OMFI component includes Feedback Metadata, Output Metadata, Quality Scores, and Recommendation Metadata. Feedback Metadata captures the details of the feedback received, Output Metadata stores information about the generated outputs, Quality Scores represent the quantitative assessments of output quality, and Recommendation Metadata encapsulates the suggestions for improvement. The main application data involved in the OMFI component includes:
- 15

Feedback Data: This data represents the feedback collected from various sources, including automated tests, user surveys, and expert reviews. It includes the feedback metadata, such as the source, timestamp, and specific comments or ratings.

- 20 Output Data: This data represents the AI-generated outputs that are subject to quality assessment and classification. It includes the output content and associated metadata.

Quality Assessment Data: This data represents the results of the quality assessments performed on the AI-generated outputs. It includes the scores for different quality criteria and the overall quality score.

- 25 Recommendation Data: This data represents the actionable recommendations generated by OMFI for improving the output quality. It includes the specific improvement actions and the associated conditions or triggers.

Functions:

OMFI incorporates the following key application functions:

Feedback Integration: This function gathers feedback from multiple sources, such as automated tests, user surveys, and expert reviews, and consolidates this feedback into a structured format for further analysis.

Quality Assessment: This function analyzes the consolidated feedback, evaluates the quality of the AI-generated outputs based on predefined criteria (e.g., accuracy, relevance, completeness, consistency), and assigns quality scores or ratings to the outputs.

Recommendation Generation: Based on the quality assessments and feedback analysis, this function generates actionable recommendations for improving the output quality. It identifies specific areas or aspects that require enhancement, prioritizes the recommendations based on potential impact and feasibility, and provides guidance for implementing the suggested improvements.

Output Classification: This function classifies the AI-generated outputs into different categories (e.g., excellent, good, needs improvement, poor, unacceptable) based on their quality scores and predefined thresholds. This classification aids in determining the appropriate course of action for each output.

Metadata Management: OMFI manages and maintains metadata related to feedback, output details, quality assessments, and recommendations. This metadata is stored in a structured format (e.g., JSON) to facilitate easy retrieval, analysis, and integration with other components of the AI system.

Feedback and iterative learning

The COGNIGEN-AX system integrates feedback from various sources and enables iterative learning and dynamic model adaptation through several key components and methods:

The OMFI (Output Monitoring and Feedback Integration) agent is responsible for processing and consolidating feedback signals from multiple modalities such as user interactions, performance metrics, automated tests, and external evaluations. OMFI employs techniques like weighted averaging to integrate feedback scores from different sources. Each source is assigned a weight (w_j) based on its perceived reliability or importance:

$$F_i = (\sum_j w_j * f_{ij}) / (\sum_j w_j)$$

F_i is the integrated feedback score for output i , and f_{ij} is the score from source j .

The weights can be manually tuned by domain experts or learned automatically based on the sources' past performance regarding feedback alignment with ground truth evaluations.

OMFI also prioritizes feedback based on multi-criteria decision analysis techniques like the Analytic Hierarchy Process (AHP), considering factors like feedback source reliability, severity of the issue identified, and relevance to the target domain/application. The integrated feedback analysis from OMFI flows into the ALMA (Adaptive Learning and Model Augmentation) agent, forming strategic reinforcement plans for enhancing the AI models.

ALMA employs case-based reasoning, ontology, and transfer learning to augment the models' knowledge bases, refine reasoning strategies, and optimize model architectures/parameters based on feedback-driven insights. It generates instructions for techniques like hyperparameter tuning, knowledge base expansion via ontology population, and even neural architecture search to adapt the models iteratively.

The IRAP (Iterative Response and Process) methodology generates updated responses incorporating the feedback/self-critique, enabling a closed reinforcement loop. The iterative improvement process continues, with ALMA analyzing the updated experience logs to detect new patterns, formulate revised plans, and further refine the models until performance goals are achieved or iteration limits are reached.

The system dynamically integrates multi-source feedback through principled techniques like weighted averaging and AHP-based prioritization in OMFI. ALMA then drives continuous adaptation by mining this feedback to devise strategic enhancements to knowledge, reasoning, and model architectures - enabling iterative learning and refinement in a structured manner.

Example:

The prompt asks you to map financial and regulatory concepts from a source dataset to a standardized metadata representation. One of the records in this dataset is:

SARB:ei93:RTG11,% Premium method ranking,ei93,RTG11, Hierarchy for ranking of methods used in provisions assumptions (enumeration),Ratings,, "Method no.1 in ranking,Method no.2 in ranking,Method no.3 in ranking,Method no.4 in ranking,Method no.5 in ranking,Method no.6 in ranking,Method no.7 in ranking,Method no.8 in ranking,Method no.9 in ranking"

After mapping the "% Premium method ranking" concept to the specified metadata attributes (Concept, Description, Meaning, Usage, etc.), the system generates an output row like:

"SARB:ei93:RTG11" | "% Premium method ranking" | "Premium method ranking" | "Ranking of the percentage premium method, which calculates provisions as a percentage of

premiums earned, among various methods used for estimating provisions." | ... | "High" | "4" | "High"

The OMFI agent collects feedback from various sources:

User interactions: Feedback from domain experts (e.g., insurance professionals) on the accuracy and relevance of the generated description, meaning, usage, etc., for the "Premium method ranking" concept.

Automated tests: Results from test cases that validate the generated output's adherence to predefined rules and constraints for mapping insurance-related concepts.

Performance metrics: Metrics like precision, recall, and F1-score computed by comparing the system's output against ground truth mappings for the "% Premium method ranking" concept.

OMFI integrates these feedback scores using the weighted average formula:

$$F_i = (w_1 * f_{i1} + w_2 * f_{i2} + w_3 * f_{i3}) / (w_1 + w_2 + w_3)$$

Where f_{i1} is the user feedback score, f_{i2} is the test case score, and f_{i3} is the performance metric score. The weights w_1 , w_2 , and w_3 are assigned based on each feedback source's perceived reliability and importance for this specific insurance domain.

For example, if user feedback from experts is considered most crucial, it could be assigned a higher weight (e.g., $w_1 = 0.5$), while test cases and metrics receive lower weights (e.g., $w_2 = 0.3$, $w_3 = 0.2$).

OMFI prioritizes feedback based on factors like reliability (e.g., reputable insurance experts), severity (e.g., feedback indicating a critical misunderstanding of the "% Premium method ranking" concept), and domain relevance (e.g., feedback specific to insurance provisioning methodologies). The integrated feedback analysis from OMFI flows into the ALMA agent, which analyzes the root causes of any issues or limitations identified in the feedback. For instance, if the feedback suggests a lack of clarity in the generated description or meaning of the "Premium method ranking" concept, ALMA could use techniques like causal graphs to trace the root causes (e.g., gaps in the knowledge base related to insurance terminology, flawed reasoning assumptions about provisioning methods).

Expanding the knowledge base:

Ontology population: ALMA will automatically employ advanced relation extraction techniques like Open Information Extraction (OpenIE) and Open Relation Extraction (ORE)

to construct a comprehensive insurance provisioning ontology. This ontology will capture intricate details about various methodologies, including the "percentage premium method," from authoritative sources like actuarial journals, regulatory guidelines, and industry white papers.

- 5 The ontology will model the "percentage premium method" as a subclass of "Provisioning Methodology," with properties capturing its calculations (e.g., provisions = fixed percentage * earned premiums), assumptions (e.g., stable loss ratios, homogeneous risk portfolios), strengths (e.g., simplicity, historical data availability), and limitations (e.g., sensitivity to premium volume changes, inability to capture risk heterogeneity).
- 10 Transfer learning: ALMA will leverage state-of-the-art language models, pre-trained on vast financial and risk management corpora, and fine-tune them on insurance-specific data using multi-task learning and domain-adaptive pretraining techniques. This will enable the effective transfer of knowledge from related domains while simultaneously specializing the models for the insurance context.
- 15 Refining reasoning strategies:
- Hierarchical relationship prioritization: ALMA will develop graph convolutional network (GCN) and graph attention network (GAT) architectures tailored to reasoning over the hierarchical structures present in the source data. These architectures will learn to encode the hierarchical relationships between concepts like "% Premium method ranking" and its
- 20 parent/child concepts, enabling more accurate mapping to the metadata attributes.
- Domain constraint integration: ALMA will incorporate a neuro-symbolic reasoning module that integrates first-order logic rules and constraints specific to the insurance domain. For instance, when mapping the "% Premium method ranking" concept, the system will prioritize fields like "Hierarchy" and "MemberLabel" based on the rule: "IF concept relates to
- 25 'Provisioning Methodology' THEN extract ranking information from 'Hierarchy' and 'MemberLabel' fields."
- Expert knowledge infusion: ALMA will develop an interactive knowledge acquisition interface for insurance experts to provide feedback, examples, and rules. This interface will leverage techniques like rationale generation and example critiquing to elicit human-understandable
- 30 explanations, which will be distilled into the reasoning strategies using meta-learning and neural program induction.

Model optimization:

Neural architecture search (NAS): ALMA will employ cutting-edge NAS techniques like Efficient Neural Architecture Search (ENAS) and Differentiable Architecture Search (DARTS) to discover optimal model configurations for insurance tasks. These methods will search over complex operation spaces, including multi-head attention, dilated convolutions, and squeeze-and-excitation blocks, while optimizing for performance on a stratified validation set covering diverse insurance concepts and hierarchies.

Hyperparameter tuning: ALMA will leverage state-of-the-art Bayesian optimization libraries like Ax and Optuna to fine-tune critical hyperparameters like learning rates, regularization strengths, and attention heads. These libraries will construct accurate surrogate models (like Gaussian Processes and Tree-structured Parzen Estimators) to efficiently explore the high-dimensional hyperparameter space, identifying configurations that maximize performance on insurance tasks while minimizing computational costs.

Ensemble methods: ALMA will investigate advanced ensemble techniques like Snapshot Ensembling and Adaptive Mixture of Experts. These methods will combine the strengths of multiple models trained on different subsets of the insurance data (e.g., personal lines vs. commercial lines) or with different inductive biases (e.g., hierarchical vs. domain-constrained), enabling the ensemble to leverage complementary "Ranking of the percentage premium method, a widely used technique in insurance provisioning that calculates reserves as a fixed percentage of earned premiums based on historical loss ratios, among various methods for estimating provisions and determining regulatory capital requirements."

Agent: MIRROR

MIRROR (Metacognitive Introspection and Reflection for Reasoned Output Refinement) Methods. MIRROR enables the AI system to engage in self-reflection on its generated prompt responses. It provides a structured approach to introspection and self-analysis, allowing the AI to critically examine outputs, identify areas for improvement, and propose actionable refinements.

Self-Analysis Generation:

MIRROR generates a comprehensive self-analysis by combining feedback from automated evaluations, user feedback, and expert reviews. It identifies potential flaws, gaps, and shortcomings in the prompt responses and underlying reasoning process. The self-analysis output is included in the self-analysis metadata, which follows the ISO 11179 metadata structure.

MIRROR uses a multi-perspective analysis approach, considering the following aspects:

Relevance: Measures cosine similarity between the prompt and response embeddings

Coherence: Analyzes the discourse structure using Rhetorical Structure Theory (RST)

Factual Accuracy: Employs Named Entity Recognition (NER) and fact-checking against knowledge bases

- 5 Completeness: Evaluates the coverage of key points using a custom completeness score:

$$C_i = (\sum_j s_{ij}) / N$$

Where C_i represents the completeness score for response i , s_{ij} is the binary indicator of whether key point j is covered in response i , and N is the total number of key points.

Strength and Weakness Identification:

- 10 MIRROR identifies the strengths and weaknesses of prompt responses based on the self-analysis output and integrated assessment. It highlights key strengths, discusses reasoning errors, inconsistencies, and inaccuracies, provides specific examples, and summarizes the main takeaways. The strengths and weaknesses are included in the strengths and weaknesses output, following the ISO 11179 metadata structure.

- 15 MIRROR utilizes a rule-based approach to identify strengths and weaknesses, with rules formulated as follows:

IF (condition_1) AND (condition_2) AND ... AND (condition_n) THEN (strength/weakness)

For example:

- IF (relevance_score > 0.8) AND (coherence_score > 0.7) THEN (strength: "The response is
20 highly relevant to the prompt and maintains a coherent discourse structure.")

Improvement Suggestion:

- MIRROR suggests modifications to enhance the prompt responses based on the self-analysis and strengths and weaknesses outputs. It proposes targeted improvements, recommends techniques to address identified weaknesses, provides illustrative examples,
25 and discusses alignment with original goals and requirements. The improvement suggestions are included in the improvement suggestions output, following the ISO 11179 metadata structure.

MIRROR generates improvement suggestions through a case-based reasoning (CBR) approach:

1. Retrieve: Locate similar past cases based on the identified strengths, weaknesses, and context.
2. Reuse: Adapt the solutions from the retrieved cases to the current situation
3. Revise: Modify the adapted solutions based on the specific requirements and constraints
- 5 4. Retain: Store the new case and its solution for future use

Application of Metacognitive Techniques:

MIRROR uses metacognitive techniques to gain insights into the decision-making process and reasoning behind the prompt responses. It employs counterfactual reasoning to explore alternative approaches, performs causal inference to understand the relationship between choices and outcomes, and engages in introspective questioning to examine assumptions, biases, and uncertainties. The findings from these metacognitive techniques are included in the metacognitive insights output, represented using the ISO 11179 metadata structure.

For counterfactual reasoning, MIRROR utilizes a structural causal model (SCM) to represent the relationships between variables:

$$Y_i = f_i(PA_i, U_i)$$

where Y_i is the outcome variable i , PA_i are the parent variables of Y_i , U_i are the exogenous variables, and f_i is the causal mechanism.

For causal inference, MIRROR employs the do-calculus to estimate the causal effect of an intervention:

$$P(Y|\text{do}(X=x)) = \sum_z P(Y|X=x, Z=z) \times P(Z=z)$$

where Y is the outcome variable, X is the intervention variable, and Z are the confounding variables.

Design: Business Layer

Actors: The business actors involved in the MIRROR component include AI Engineers, Data Scientists, and Domain Experts. AI Engineers are responsible for developing and maintaining the AI system, Data Scientists analyze and interpret the self-reflection outputs, and Domain Experts provide subject matter expertise to validate the insights and recommendations.

Services

MIRROR offers the business service of Introspective Analysis and Self-Reflection. This service enables the AI system to critically examine its outputs, reasoning processes, and decision-making rationales, facilitating continuous improvement and refinement of the generated results.

5 Processes

MIRROR facilitates self-reflection and iterative refinement as its primary business processes. This process involves generating self-analyses of the AI system's outputs, identifying strengths and weaknesses, proposing targeted improvements, and employing metacognitive techniques to gain deeper insights into the reasoning process. These insights are then used to iteratively refine and optimize the AI system's performance.

Functions:

MIRROR supports the business functions of Output Quality Assurance, Reasoning Transparency, and Continuous Learning. It ensures the quality of AI-generated outputs by enabling self-evaluation and introspection, promotes transparency in the reasoning process by surfacing decision rationales and assumptions, and facilitates continuous learning by incorporating self-reflective insights into the system's knowledge base and adaptation strategies.

Objects:

The MIRROR component in the application layer involves key business objects such as AI Outputs, Integrated Assessments, Self-Reflection Metadata, and Improvement Recommendations. AI Outputs are the generated results or responses. Integrated Assessments represent the consolidated evaluation of output quality. Self-reflection metadata captures insights and analysis from the reflective process. Improvement Recommendations provide actionable suggestions for enhancing the outputs based on self-reflection.

The MIRROR component, known as Metacognitive Introspection and Reflection for Reasoned Output Refinement, is the core application component.

MIRROR provides the following application services: Self-Analysis Service, Strength and Weakness Identification Service, Improvement Suggestion Service, and Metacognitive Reasoning Service. These services facilitate the generation of self-analyses, identification of strengths and weaknesses, proposal of targeted improvements, and application of

metacognitive techniques (such as counterfactual reasoning, causal inference, and introspective questioning), respectively.

MIRROR exposes several application interfaces, including the Output Analysis Interface for ingesting AI outputs and associated metadata, the Knowledge Base Interface for accessing domain knowledge and reasoning rules, and the Reporting and Visualization Interface for presenting self-reflection insights and recommendations.

The data used within the MIRROR component includes Output Metadata, Integrated Assessments, Self-Reflection Metadata, Metacognitive Analysis Results, and Improvement Recommendations. Output Metadata stores information about the generated outputs, Integrated Assessments represent the consolidated evaluation of output quality, Self-Reflection Metadata captures the textual self-analyses and identified strengths and weaknesses, Metacognitive Analysis Results encapsulate the findings from applying metacognitive techniques, and Improvement Recommendations provide actionable suggestions for enhancing the outputs.

MIRROR incorporates key application functions:

Self-Analysis Generation: This function generates comprehensive self-analyses of the AI system's outputs, identifying potential flaws, gaps, and shortcomings. It analyzes errors, inconsistencies, and suboptimal results, discussing their origins and propagation, and summarizing key findings and insights.

Strength and Weakness Identification: Based on the self-analysis and integrated assessments, this function highlights the strengths and weaknesses of the AI-generated outputs. It identifies key strengths, discusses reasoning errors, inconsistencies, and inaccuracies, provides specific examples, and summarizes the main takeaways.

Improvement Suggestion: Based on self-analysis and identification of strengths and weaknesses, this function proposes enhancements to improve AI-generated outputs. It suggests specific modifications, provides techniques to address weaknesses, includes illustrative examples, and discusses alignment with original goals and requirements.

Metacognitive Technique Application: This function utilizes advanced metacognitive techniques to gain insights into the reasoning and rationale behind AI-generated outputs. It employs counterfactual reasoning to explore alternative approaches, applies causal inference to understand the relationship between choices and outcomes, and engages in introspective questioning to examine assumptions, biases, and uncertainties.

Self-Reflection Metadata Generation: This function generates structured metadata that represents the self-reflection process, including self-analysis, identified strengths and weaknesses, proposed improvement suggestions, and metacognitive insights. The metadata is stored in a standardized format (e.g., JSON) for future reference and analysis.

5 CERES (Contextual Encoding and Reflective Experience Structuring) Methods:

CERES is a method that encodes and structures the various components involved in the prompt-response cycle of a generative AI system. It ensures a comprehensive representation of the AI's experience by encoding the task prompt, initial outputs, evaluative feedback, and self-reflective analyses into a structured data format that follows the ISO 11179 metadata standard.

Context Encoding:

CERES captures and encodes the original context of the task, including the task prompt, parsed requirements (input and output specifications, task type, and skill domain), and the domain context. The encoded task context is represented in accordance with the ISO 11179 metadata structure, ensuring standardization and interoperability.

CERES utilizes a hierarchical encoding approach to represent the task context:

Parsed Requirements:

Input Specifications: Encoded using a semantic parsing model (e.g., Abstract Meaning Representation) to generate a structured representation.

Output Specifications: Encoded using a semantic parsing model (e.g., Abstract Meaning Representation) to generate a structured representation.

Task Type: Encoded using a one-hot vector representation, based on a predefined taxonomy of task types.

Skill Domain: Encoded using a multi-label classification model (e.g., sigmoid activation with binary cross-entropy loss) to generate a probability distribution across skill domains.

Output Encoding:

CERES captures the initial outputs produced by the AI system, as well as the reasoning artifacts such as retrieved knowledge, inference steps, and uncertainty estimates. To provide additional context and traceability, it includes metadata related to the outputs, such as output type, format, and timestamp. The encoded initial outputs and reasoning artifacts are represented in accordance with the ISO 11179 metadata structure.

CERES employs the following encoding techniques for output encoding:

Reasoning Artifacts:

Retrieved Knowledge: Encoded using a knowledge graph embedding model to generate a dense vector representation for each knowledge triple.

- 5 Inference Steps: Encoded using a sequence-to-sequence model to generate a dense vector representation for each inference step.

Uncertainty Estimates: Encoded using a probability distribution over possible outputs, generated by a Bayesian neural network or a model with dropout layers.

Output Metadata:

- 10 Output Type: Encoded using a one-hot vector representation, based on a predefined taxonomy of output types.

Output Format: Encoded using a one-hot vector representation, based on a predefined taxonomy of output formats.

Output Timestamp: Encoded using a normalized scalar value that represents the timestamp.

- 15 Feedback Encoding: CERES captures evaluative feedback, including feedback metadata (source and timestamp) and quality assessment (evaluation metrics and error analysis), on the initial outputs. The encoded evaluative feedback is represented using the ISO 11179 metadata structure to ensure consistency and facilitate analysis.

For feedback encoding, CERES utilizes the following techniques:

- 20 Feedback Metadata:

Feedback Source: Encoded using a one-hot vector representation based on a predefined taxonomy of feedback sources.

Feedback Timestamp: Encoded using a normalized scalar value representing the timestamp.

- 25 Quality Assessment:

Evaluation Metrics: Encoded using a vector of normalized scalar values, each representing the score for a specific evaluation metric (e.g., accuracy, precision, recall, F1-score).

Error Analysis: Encoded using a multi-label classification model (e.g., sigmoid activation with binary cross-entropy loss) to generate a probability distribution over error types.

- 30 CERES also captures the AI system's textual self-reflection, including identified strengths, weaknesses, and insights from the experience. It encodes metacognitive analyses, such as counterfactual reasoning, causal inference, and uncertainty assessment, to provide deeper

insights into the AI system's decision-making process. The encoded self-reflection and metacognitive analyses are represented using the ISO 11179 metadata structure.

For self-reflection encoding, CERES employs the following techniques:

Self-Reflection Text:

5 Strengths: Encoded to generate a dense vector representation.

Weaknesses: Encoded to generate a dense vector representation.

Insights: Encoded to create a dense vector representation.

Metacognitive Analyses:

10 Counterfactual Reasoning: Encoded using a causal inference model (e.g., structural causal model) to generate a dense vector representation for each counterfactual scenario.

Causal Inference: Encoded using a causal inference model (e.g., causal Bayesian network) to generate a dense vector representation for each causal relationship.

Uncertainty Assessment: Encoded using a probability distribution over possible outcomes, generated by a Bayesian neural network or a model with dropout layers.

15 To establish meaningful relationships and enable efficient navigation and retrieval of the encoded experience, CERES creates semantic links between the encoded components.

These links connect tasks and outputs, outputs and feedback, reflection and outputs, reflection and feedback, and capture the dependencies and flow of information. Semantic links are represented using standard formats such as RDF (Resource Description

20 Framework) or JSON-LD (JSON for Linking Data), adhering to the ISO 11179 standard.§

Task-Output Links: These links are generated using a semantic similarity model (for example, a Siamese neural network) to identify the relationships between task contexts and outputs.

25 Output-Feedback Links: These links are generated using a semantic similarity model (for example, a Siamese neural network) to identify the relationships between outputs and feedback.

Reflection-Output Links: These links are generated using a semantic similarity model (for example, a Siamese neural network) to identify the relationships between self-reflection and outputs.

30 Reflection-Feedback Links: These links are generated using a semantic similarity model (for example, a Siamese neural network) to identify the relationships between self-reflection and feedback.

Provenance Links: These links are generated using a provenance tracking model (for example, a directed acyclic graph) to capture the dependencies and flow of information between components.

Experience Encoding Generation:

5 CERES combines all the encoded components (such as task context, initial outputs, evaluative feedback, self-reflection, and semantic links) into a coherent and structured experience encoding. It ensures that the generated encoding adheres to the ISO 11179 metadata standard for consistency, completeness, and correctness. The experience encoding is validated to ensure its integrity and compliance with the standard.

10 CERES uses the following techniques for experience encoding generation:

Component Integration: It concatenates the encoded representations of task context, initial outputs, evaluative feedback, self-reflection, and semantic links into a single vector representation.

15 Metadata Validation: It validates the generated experience encoding against the ISO 11179 metadata standard using a schema validation model (such as a JSON Schema validator).

Consistency Checking: It checks the consistency of the generated experience encoding using a consistency checking model (such as a knowledge graph consistency checker) to ensure logical coherence and avoid contradictions.

20 Completeness Checking: It checks the completeness of the generated experience encoding using a completeness checking model (such as an ontology completeness checker) to ensure that all required components are present.

Correctness Checking: It checks the correctness of the generated experience encoding using a correctness checking model (such as a fact-checking model) to ensure the accuracy of the encoded information.

25 CERES Business Layer:

Actors

The key business actors involved in the CERES component include AI Engineers, Data Architects, and Knowledge Managers. AI Engineers are responsible for developing and maintaining the AI system. Data Architects ensure the proper structuring and encoding of
30 data and metadata. Knowledge Managers oversee the management and curation of the AI system's knowledge base.

Services: CERES is a business service that offers Experience Encoding and Knowledge Base Management. It helps capture, structure, and encode the AI system's experiences, including task contexts, outputs, feedback, and self-reflections, into a standardized format. It also supports integrating these experiences into the system's knowledge base.

- 5 A process facilitated by CERES is Experience Encoding and Knowledge Base Enrichment. This involves encoding the various components of the AI system's prompt-response cycle, creating semantic links between them, and generating a comprehensive experience encoding that follows industry standards. These encoded experiences are then integrated into the knowledge base to enhance the system's learning.
- 10 CERES supports the business functions of Data Integration, Knowledge Management, and Provenance Tracking. It ensures seamless integration of data and metadata from different sources, facilitates effective management of the AI system's knowledge base, and enables tracking of provenance and dependencies within the encoded experiences.

- The key objects in the CERES component are Task Contexts, AI Outputs, Feedback Metadata, Self-Reflection Metadata, Semantic Links, and Experience Encodings. Task Contexts represent the original prompts and requirements, AI Outputs are the system's generated results or responses, Feedback Metadata captures evaluations and quality assessments, Self-Reflection Metadata encapsulates introspective analyses, Semantic Links establish relationships between components, and Experience Encodings are structured representations of the system's experiences.
- 15
 - 20

CERES, which stands for Contextual Encoding and Reflective Experience Structuring, is the core application component.

- CERES provides several application services, including the Context Encoding Service, Output Encoding Service, Feedback Encoding Service, Self-Reflection Encoding Service, Semantic Linking Service, and Experience Encoding Generation Service. These services facilitate encoding task contexts, AI outputs, feedback, self-reflections, establishing semantic links, and generating comprehensive experience encoding. CERES provides several application interfaces, each serving a different purpose. The Task Context Interface is used for inputting prompts and requirements, the Output Interface receives AI-generated outputs, the Feedback Interface captures evaluations and quality assessments, the Self-Reflection Interface ingests introspective analyses, and the Knowledge Base Interface integrates encoded experiences into the system's knowledge base.
- 25
 - 30

Within the CERES component, various types of data are used. This includes Task Context Metadata, Output Metadata, Feedback Metadata, Self-Reflection Metadata, Semantic Link Metadata, and Experience Encoding. Task Context Metadata captures the original prompts and requirements, Output Metadata stores information about the generated outputs,
5 Feedback Metadata encompasses evaluations and quality assessments, Self-Reflection Metadata represents introspective analyses, Semantic Link Metadata establishes relationships between components, and Experience Encoding is the comprehensive, structured representation of the AI system's experience.

CERES incorporates several key application functions to fulfill its purpose. Context Encoding
10 captures and encodes the original task context, including prompts, requirements, and domain context. Output Encoding encodes the initial outputs generated by the AI system, as well as the associated reasoning artifacts. Feedback Encoding encodes the evaluative feedback received on the initial outputs. Self-Reflection Encoding encodes the AI system's textual self-reflection, including weaknesses and insights. Semantic Linking establishes
15 meaningful relationships between encoded components, enabling easy navigation and retrieval of the encoded experience.

Experience Encoding Generation: This function takes all the encoded components, including task context, initial outputs, evaluative feedback, self-reflection, and semantic links, and combines them into a well-organized and structured experience encoding. This encoding
20 follows the ISO 11179 metadata standard, ensuring consistency, completeness, and correctness. The resulting encoding is provided in a structured format, such as JSON or XML, for easy storage and future reference.

Agent: ALMA

ALMA (Adaptive Learning and Model Augmentation) analyzes accumulated experience logs
25 to identify patterns and opportunities for improvement. It develops strategic reinforcement plans to enhance the AI system's performance by leveraging historical data and fine-tuning cognitive systems and methods.

Experience Log Analysis:

ALMA analyzes experience logs by pre-processing the data and applying statistical analysis,
30 pattern mining, and machine learning techniques. This analysis helps identify trends, patterns, and anomalies, resulting in actionable insights.

ALMA utilizes the following techniques for experience log analysis:

Data Pre-processing:

Data Cleaning: Removes noise, outliers, and inconsistencies from experience logs through techniques such as filtering, smoothing, and normalization.

- 5 Data Integration: Combines experience logs from multiple sources and resolves conflicts using techniques like data fusion and record linkage.

Data Transformation: Converts the experience logs into a suitable format for analysis through techniques such as feature scaling, one-hot encoding, and dimensionality reduction (e.g., PCA, t-SNE).

Statistical Analysis:

- 10 Descriptive Statistics: Computes summary statistics like mean, median, mode, standard deviation, and quartiles to understand the distribution and central tendencies of experience log data.

Time Series Analysis: Analyzes temporal patterns and trends in experience logs using techniques such as moving averages, exponential smoothing, and ARIMA models.

- 15 Pattern Mining:

Association Rule Mining: Discovers frequent itemsets and association rules in experience logs using algorithms like Apriori and FP-Growth.

Sequential Pattern Mining: Identifies frequent subsequences and sequential patterns in experience logs using algorithms like PrefixSpan and SPADE.

- 20 Machine Learning:

Clustering: Groups similar experiences together based on their features using algorithms like K-means, hierarchical clustering, and DBSCAN.

Anomaly Detection: Identifies unusual or rare experiences that deviate from the norm using techniques like one-class SVM, isolation forest, and autoencoders.

- 25 The findings from experience log analysis are summarized in the analysis output, represented using the ISO 11179 metadata structure.

Reinforcement Plan Formulation:

Based on the analysis output, ALMA formulates reinforcement plans to adapt reasoning strategies, expand knowledge bases, improve output quality, and enhance metacognitive

capabilities. The reinforcement plans are prioritized based on their potential impact and feasibility.

ALMA employs the following techniques for reinforcement plan formulation:

Reasoning Strategy Adaptation:

- 5 Rule-Based Adaptation: Modifies the reasoning process by adjusting the rules and heuristics based on patterns and insights found in the analysis of the experience log.

Case-Based Adaptation: Retrieves similar past experiences and applies their solutions to the current situation using techniques such as case retrieval, case reuse, and case revision.

Knowledge Base Expansion:

- 10 Ontology Learning: Automatically acquires new concepts, relationships, and hypotheses from the experience logs and integrates them into the existing ontology using methods like association rule mining and inductive logic programming.

Knowledge Graph Completion: Predicts missing links and attributes in the knowledge graph based on observed patterns in the experience logs using techniques such as embedding-

- 15 based models and graph neural networks.

Output Quality Improvement:

Natural Language Generation: Enhances the fluency, coherence, and diversity of generated outputs using techniques like template-based generation, neural language models, and reinforcement learning.

- 20 Error Correction: Identifies and corrects errors in the generated outputs using techniques such as grammar correction, spelling correction, and checking for semantic consistency.

Metacognitive Capability Enhancement:

Introspection: Enhances the AI system's ability to monitor and analyze its own reasoning process using techniques such as attention mechanisms, explainable AI, and self-

- 25 supervised learning.

Uncertainty Estimation: Enhances the AI system's ability to quantify and communicate the uncertainty associated with its predictions and decisions using techniques such as Bayesian neural networks, ensemble methods, and calibration techniques.

The detailed reinforcement plans are stored in the metadata for reinforcement plans, using the ISO 11179 metadata structure to ensure consistent representation and facilitate action.

Updated Component Provisioning:

- 5 ALMA generates instructions to update models and components based on the reinforcement plans. It incorporates adaptations, fine-tunes models, expands knowledge bases, and adjusts configurations and hyperparameters.

ALMA employs the following techniques for updated component provisioning:

Model Fine-Tuning:

- 10 Transfer Learning: Adapts pre-trained models to new tasks or domains by adjusting their weights using techniques such as gradient descent and regularization.

Hyperparameter Optimization: Searches for the optimal hyperparameters of the models using techniques such as grid search, random search, and Bayesian optimization.

Knowledge Base Expansion:

- 15 Ontology Population: Adds new instances and facts to the existing ontology based on insights from the analysis of the experience log using techniques such as named entity recognition and relation extraction.

Knowledge Graph Embedding: Learns low-dimensional vector representations of entities and relations in the knowledge graph for efficient reasoning and inference.

Configuration Adjustment:

- 20 Parameter Tuning: Adjusts AI system's component parameters based on reinforcement plans using techniques like sensitivity analysis and automated parameter optimization.

Architectural Modification: Modifies the architecture of AI system's components based on reinforcement plans using techniques like neural architecture search and evolutionary algorithms.

- 25 Detailed instructions for updating models and components are provided in the updated components metadata, represented using the ISO 11179 metadata structure for consistent representation and easy implementation.

Cognitive Pipeline Customization:

ALMA provides instructions for customizing the cognitive pipeline for each task. It retrieves relevant experiences and components, calibrates parameters, and determines self-reflection and feedback mechanisms.

ALMA employs the following techniques for cognitive pipeline customization:

5 Experience Retrieval:

Case-Based Reasoning: Retrieves most relevant past experiences based on similarity to current task context using techniques like k-nearest neighbors and cosine similarity.

Semantic Search: Finds relevant experiences and components based on semantic similarity to current task using techniques like latent semantic analysis and word embeddings.

10 Parameter Calibration:

Bayesian Optimization: Adjusts cognitive pipeline component parameters based on task requirements and constraints using techniques like Gaussian processes and expected improvement.

15 Sensitivity Analysis: Assesses impact of different parameter settings on cognitive pipeline performance using techniques like local and global sensitivity analysis.

Self-Reflection Mechanism:

Metacognitive Questioning: Generates self-reflective questions to guide introspection process based on task context and performance using techniques like template-based question generation and natural language understanding.

20 Uncertainty-Aware Reasoning: Incorporates uncertainty estimates into self-reflection process for more robust and calibrated introspection using techniques like probabilistic graphical models and Bayesian inference.

Feedback Incorporation:

25 Active Learning: Selects most informative experiences for feedback based on potential to improve cognitive pipeline using techniques like uncertainty sampling and query-by-committee.

Reinforcement Learning: Adapts cognitive pipeline based on received feedback using techniques like Q-learning and policy gradients.

ALMA Metadata Creation:

ALMA creates structured metadata that represents the ALMA process. This includes the analysis of experience logs, strategic reinforcement plans, updated components, customized cognitive pipeline, and a summary of key takeaways. The ALMA metadata is output in JSON format using the ISO 11179 metadata structure to ensure standardization, interoperability, and facilitate data exchange and analysis.

ALMA Business Layer

Actors:

The critical business actors involved in the ALMA component include AI Engineers, Machine Learning Engineers, and Data Scientists. AI Engineers are responsible for developing and maintaining the AI system. Machine Learning Engineers oversee the training and optimization of the AI models. Data Scientists analyze the experience logs and historical data to derive insights for system adaptation.

Services:

ALMA offers the business service of Adaptive Learning and Model Optimization. This service enables the AI system to continuously learn and adapt based on its experiences. It leverages historical data and feedback to fine-tune its models, optimize performance, and align with evolving user needs and requirements.

Processes:

The primary business process facilitated by ALMA is Experience Analysis and Continuous Adaptation. This process involves analyzing the experience logs generated by the AI system. It identifies patterns, trends, and areas for improvement. It formulates strategic reinforcement plans to enhance the system's performance. These plans are then implemented through model updates, hyperparameter tuning, and cognitive pipeline customization. This enables the AI system to adapt and evolve continuously.

Functions:

ALMA supports the business functions of Model Governance, Performance Monitoring, and Continuous Improvement. It ensures proper governance and oversight of the AI models. It enables transparent and accountable model updates and changes. Additionally, it facilitates continuous monitoring of the system's performance. It identifies potential issues or degradation and drives iterative improvements to maintain and enhance AI system effectiveness.

Objects:

The key business objects involved in the ALMA component include Experience Logs, Historical Data, Reinforcement Plans, Model Artifacts, and Performance Metrics. Experience Logs capture the AI system's experiences and interactions. Historical Data represents the accumulated data from previous iterations. Reinforcement Plans outline the strategies for system adaptation. Model Artifacts encapsulate the trained AI models and their configurations. Performance Metrics provide quantitative measures of the system's effectiveness.

ALMA Application Layer:

10 The core application component is ALMA (Adaptive Learning and Model Augmentation).

ALMA provides the following application services:

1. Experience Log Analysis Service: Analyzes experience logs, extracting relevant information and structuring the data for further analysis and pattern detection.
2. Pattern Detection Service: Uses advanced analytical techniques to identify patterns, trends, and opportunities for improvement within the experience logs and historical data.
3. Reinforcement Planning Service: Formulates strategic reinforcement plans based on the identified patterns and insights to enhance the system's performance.
4. Model Update Service: Implements the reinforcement plans by updating AI models, fine-tuning hyperparameters, and customizing cognitive pipelines.
5. Performance Monitoring Service: Continuously monitors the system's performance and provides quantitative measures of its effectiveness.

ALMA exposes several application interfaces:

1. Experience Log Interface: Ingests the AI system's experience logs.
2. Historical Data Interface: Accesses accumulated data from previous iterations.
3. Model Artifact Interface: Retrieves and updates trained AI models.
4. Reporting and Visualization Interface: Presents performance metrics and adaptation insights.

The application data used within the ALMA component includes:

1. Experience Logs: Capture the AI system's experiences and interactions.
2. Historical Data: Represents accumulated data from previous iterations.
3. Reinforcement Plans: Outline strategies for system adaptation.
4. Model Artifacts: Encapsulate trained AI models and their configurations.
5. Performance Metrics: Provide quantitative measures of the system's effectiveness.

ALMA incorporates the following key application functions:

1. Experience Log Analysis: Analyzes experience logs, extracting relevant information for further analysis.
2. Pattern Detection: Identifies patterns, trends, and opportunities for improvement within the experience logs and historical data.
3. Reinforcement Planning: Formulates strategic plans to enhance the system's performance based on identified patterns and insights.
4. Model Update: Implements reinforcement plans by updating AI models and optimizing their performance.
- 15 Performance Monitoring is a function that constantly tracks various metrics, including accuracy, precision, recall, and user satisfaction, to assess the performance of the AI system. It establishes baselines, detects any degradation or deviations in performance, and initiates corrective actions or adaptation cycles as needed.

IPASE

- 20 IPASE (Introspective Performance Analysis and Self-Enhancement) is a methodology that helps AI systems, specifically large language models (LLMs), improve the quality and reliability of their outputs through structured self-critique and iterative improvement. By engaging in a systematic process of self-reflection, critical analysis, and enhancement, AI systems using IPASE can achieve higher quality outputs.
- 25 The key features of IPASE include:
 1. MACQL (Modularized Analysis, Quantification, Comparison, and Limitations): IPASE conducts a detailed self-critique by breaking down the AI system's initial response into modules such as relevance, coherence, factual accuracy, completeness, and style. Using quantifiable metrics, the output quality is objectively assessed, and the response is

compared to reference materials to identify any gaps. The system evaluates its response in the broader context of the task and acknowledges its limitations.

2. Root Cause Analysis: IPASE performs root cause analysis for each limitation or area for improvement. This analysis aims to identify the contributing factors such as missing knowledge, faulty assumptions, or biases that influence the response. Structured representations, like causal graphs, are used to visualize the interrelationships between these factors.

3. Iterative Improvement: Based on the self-critique and root cause analysis, IPASE generates an updated response that addresses the limitations and incorporates improvements. This process involves retrieving relevant information, revising reasoning, adapting language and style, and leveraging insights from previous iterations.

The agents involved in the IPASE process are:

MACQ: Conducts modularized analysis, quantifies output quality, compares outputs to references, and identifies limitations to ensure consistency and quality.

RAINS: Performs root cause analysis, explores knowledge and skill gaps, analyses assumptions and biases, generates causal graphs, and synthesizes actionable insights for improvement.

IRAP: Iteratively refines responses by incorporating feedback, adapting reasoning processes, updating language and style, and enabling continuous enhancement.

The evaluation and validation of IPASE use specific methodologies and objectives:

MACQ is evaluated using benchmarking datasets, expert-annotated corpora, and reference outputs. Metrics such as modular evaluation accuracy (relevance, coherence, factuality, completeness), comparative analysis precision and recall, and limitations detection performance are used.

RAINS relies on curated root cause datasets, simulated scenarios with injected faults, and expert-validated causal models. Metrics are used to assess root cause recovery rate, accuracy in mapping knowledge gaps, recall in assumption and bias detection, and fidelity in constructing causal graphs.

The goal of IPASE is to establish rigorous protocols for conducting modularized self-critiques and evaluating the outputs of AI reasoners. This involves developing standardized methods and guidelines for breaking down the output into distinct components or aspects, such as relevance, coherence, factual accuracy, completeness, and style. Each element has clear

evaluation criteria and metrics that enable objective and consistent assessment of the quality of the output. IPASE provides clear instructions for applying these evaluation protocols, ensuring that the self-critique process is systematic, comprehensive, and reproducible. By taking a modularized approach, IPASE allows AI reasoners to analyse their outputs in detail, identifying specific strengths and weaknesses that can guide targeted improvements.

Enable root cause analysis and targeted remediation planning:

IPASE enables AI reasoners to conduct root cause analysis on identified limitations or areas for improvement in their outputs. This involves developing techniques and methodologies for systematically investigating the underlying factors that contribute to suboptimal performance or errors. IPASE guides the formulation of hypothesis-driven questions, the design of diagnostic experiments, and the collection of relevant data to support the root cause analysis process. By uncovering the fundamental causes of issues, AI reasoners can gain a better understanding of their limitations and make informed decisions about strategies for improvement. IPASE includes mechanisms for generating targeted remediation plans based on the findings of the root cause analysis. These plans outline specific actions, modifications, or interventions to address the identified issues and improve the quality of the AI reasoner's output.

Quantitative evaluation and comparative analysis:

IPASE incorporates quantitative evaluation and comparative analysis capabilities to support rigorous self-critique and assessment. It develops metrics, scoring systems, and statistical methods for assessing the quality and performance of AI reasoners' outputs. IPASE provides standardized evaluation rubrics and benchmarks for consistent and objective measurement of output characteristics, such as accuracy, completeness, coherence, and style. Additionally, it includes mechanisms for comparing the AI reasoner's outputs against reference standards, domain-specific guidelines, or human-generated exemplars. Through comparative analysis, AI reasoners can identify gaps, discrepancies, or areas where their outputs deviate from the desired quality or performance levels. These capabilities generate actionable insights and recommendations for improvement, guiding AI reasoners toward achieving higher output quality standards.

IPASE enables efficient organization, retrieval, and utilization of self-critique and evaluation results. It develops standardized formats and schemas for capturing and storing the outputs of modularized evaluations, root cause analyses, and remediation plans. IPASE provides guidelines for creating well-structured and machine-readable artifacts with relevant metadata, such as timestamps, version information, and provenance details. Additionally, it

incorporates mechanisms for tracking the relationships and dependencies between different analysis artifacts, such as linking specific evaluation results to corresponding root causes and remediation actions. By maintaining a structured repository of analysis artifacts, AI reasoners can easily access and leverage previous findings to inform ongoing self-improvement efforts, ensuring continuity and consistency in the iterative refinement process.

IPASE enables AI reasoners to incorporate diverse perspectives and real-world insights into their self-critique and evaluation processes. It develops interfaces and protocols for collecting and processing feedback from external sources, such as domain experts, end-users, or independent auditors. IPASE provides mechanisms for soliciting targeted feedback on specific aspects of the AI reasoner's outputs, such as relevance, usability, or fairness. Additionally, it includes techniques for aggregating and analysing external feedback and identifying common themes, patterns, or issues that may not be apparent through internal self-critique alone. By integrating external validation and user feedback, AI reasoners can validate their assessments, identify blind spots or biases, and better understand the real-world impact and effectiveness of their outputs.

Agent MACQ

MACQ includes quantifiable metrics to objectively assess the quality of output for each module. These metrics provide a standardized way to measure performance and identify areas for improvement.

Comparison to Reference Materials:

To further evaluate the initial response, MACQ compares it to relevant reference materials. This comparison helps identify any discrepancies, gaps, or areas where the response can be enhanced.

Evaluation within Task Context:

MACQ goes beyond just analyzing the initial response and considers how it fits into the broader context of the task. It assesses any limitations, uncertainties, or potential biases that may affect the overall quality of the output. This holistic evaluation ensures that the AI system acknowledges and addresses these factors.

In summary, MACQ employs modularized analysis to break down the self-critique into distinct modules for examination. It utilizes various techniques such as semantic similarity, topic modeling, discourse parsing, coreference resolution, named entity recognition, fact-checking, information extraction, coverage assessment, readability metrics, and sentiment

analysis. Additionally, MACQ incorporates quantifiable metrics and compares the initial response to reference materials while also evaluating it within the task context. This comprehensive approach improves clarity, flow, and the overall quality of the AI system's output. MACQ generates quantifiable metrics for each critique module, providing an objective measure of the quality of the output. These metrics include relevance scores, coherence indices, factual accuracy percentages, completeness ratings, and numerical style ratings.

To quantify these metrics, MACQ employs the following techniques:

Relevance Metrics:

- 10 Semantic Similarity Scores: Compute relevance scores between 0 and 1 by calculating cosine similarity, Jaccard similarity, or embedding-based similarity between the initial response and the input query.

- 15 Topic Coherence Scores: Measure the coherence of the topics extracted from the initial response using metrics like UCI coherence, UMass coherence, or normalized pointwise mutual information (NPMI).

Coherence Metrics:

- Discourse Coherence Scores: Quantify the coherence of the discourse structure using metrics such as the number of rhetorical relations, the depth of the discourse tree, or the ratio of satellite to nucleus units.
- 20 Coreference Resolution Accuracy: Compute the accuracy of coreference resolution using metrics like precision, recall, or F1 score.

Factual Accuracy Metrics:

- Named Entity Accuracy: Calculate the accuracy of named entity recognition using metrics like precision, recall, or F1 score.
- 25 Fact Verification Accuracy: Measure the accuracy of fact verification using metrics like the percentage of correctly verified facts or the ratio of true positives to false positives.

Completeness Metrics:

- Information Coverage Scores: Quantify the coverage of key information in the initial response using metrics like the percentage of input query aspects addressed or the ratio of extracted information to total information.
- 30

Completeness Ratings: Assign numerical ratings (e.g., on a scale of 1 to 5) to the completeness of the initial response based on predefined criteria.

Style Metrics:

5 Readability Scores: Compute numerical readability scores using formulas like Flesch-Kincaid grade level, Gunning fog index, or SMOG index.

Sentiment Scores: Quantify the sentiment of the initial response using metrics like sentiment polarity (positive, negative, neutral) or sentiment intensity (e.g., on a scale of -1 to 1).

10 MACQ then compares the initial response to relevant reference materials to identify discrepancies, gaps, or areas for improvement. It categorizes the observed differences and quantifies the degree of divergence using the following techniques:

Reference Material Selection:

Domain-Specific Knowledge Bases: Select relevant domain-specific knowledge bases, ontologies, or taxonomies for comparison based on the input query and the initial response.

15 Exemplar Responses: Retrieve exemplar responses from a repository of high-quality, human-generated responses for comparison.

Discrepancy Identification:

Semantic Difference Calculation: This step calculates the semantic difference between the initial response and the reference materials. Techniques like semantic similarity measures or semantic role labeling can be used for this analysis.

20 Structural Difference Analysis: This step analyzes the structural differences between the initial response and the reference materials. Techniques like tree edit distance or graph isomorphism are used for this analysis.

Gap Analysis:

25 Missing Information Detection: This step identifies any missing information or aspects in the initial response compared to the reference materials. Techniques like set difference or information extraction are employed for this task.

Incomplete Reasoning Identification: This step detects any incomplete reasoning or logical gaps in the initial response compared to the reference materials. Techniques like argument mining or causal reasoning are used for this analysis.

Divergence Quantification:

Semantic Divergence Scores: This step quantifies the semantic divergence between the initial response and the reference materials. Metrics like cosine distance, Kullback-Leibler divergence, or Jensen-Shannon divergence are used for this quantification.

- 5 Structural Divergence Scores: This step measures the structural divergence between the initial response and the reference materials. Metrics like tree edit distance, graph edit distance, or maximum common subgraph are employed for this measurement.

Contextual Evaluation:

- 10 MACQ evaluates the initial response within the broader context of the task. Factors like the intended audience, core purpose, and stated or implied constraints are considered. It assesses the alignment of the response's content, complexity, tone, and style with these contextual factors.

MACQ utilizes the following techniques for contextual evaluation:

Audience Analysis:

- 15 User Profiling: This step analyses the characteristics, preferences, and expertise level of the intended audience. Techniques like user modeling or collaborative filtering can be used for this analysis.

- 20 Readability Assessment: This step assesses the readability and comprehensibility of the initial response for the intended audience. Readability metrics and language complexity measures are employed for this assessment.

Purpose Analysis:

Intention Classification: This step classifies the core purpose or intention of the input query. Techniques like rule-based systems, machine learning models, or natural language understanding can be used for this classification.

- 25 Goal-Response Alignment: This step measures the alignment between the initial response and the classified purpose. Metrics like precision, recall, or F1 score are used for this alignment measurement.

Constraint Analysis:

Constraint Extraction: This step extracts any stated or implied constraints from the input query. Techniques like named entity recognition, dependency parsing, or semantic role labeling can be used for this extraction.

- 5 Constraint Satisfaction Assessment: This step assesses the satisfaction of the extracted constraints in the initial response. Techniques like constraint programming or satisfiability solving are employed for this assessment.

Limitations Identification: MACQ explicitly acknowledges any limitations, uncertainties, or potential biases in its initial response. It also identifies areas where additional information or clarification may be needed.

- 10 To identify limitations, MACQ employs the following techniques:

Uncertainty Detection:

Hedging and Speculation Identification: Using rule-based systems or machine learning models trained on linguistic cues, MACQ detects hedging and speculative language in the initial response.

- 15 Confidence Estimation: MACQ estimates the confidence or certainty of the statements in the initial response using techniques such as probabilistic models, Bayesian networks, or uncertainty quantification methods.

Bias Detection:

- 20 Demographic Bias Analysis: MACQ analyzes the initial response for potential demographic biases (e.g., gender, race, age) using techniques such as sentiment analysis, word embeddings, or fairness metrics.

Ideological Bias Analysis: Using techniques such as stance detection, argument mining, or political ideology classification, MACQ detects potential ideological biases in the initial response.

- 25 Information Sufficiency Assessment:

Clarity Evaluation: MACQ assesses the clarity and specificity of the information provided in the initial response using techniques such as specificity measures or ambiguity detection.

- 30 Relevance Evaluation: MACQ evaluates the relevance and appropriateness of the information provided in the initial response using techniques such as relevance feedback or information gain metrics.

For critique synthesis, MACQ combines the modularized analysis, quantified metrics, comparative analysis, contextual evaluation, and identified limitations into a comprehensive,

well-structured self-critique report. It presents the key findings and observations in a logical flow, utilizing quantitative metrics as evidence to support qualitative assessments.

The techniques employed by MACQ for critique synthesis are as follows:

Report Structuring:

- 5 Modular Organization: The self-critique report is organized into distinct sections corresponding to each module of analysis (e.g., relevance, coherence, factual accuracy, completeness, style).

- 10 Hierarchical Information Presentation: The information in the self-critique report is presented using a hierarchical structure, with high-level summaries followed by detailed analyses and supporting evidence.

Key Findings Summarization:

Abstractive Summarization: MACQ generates concise summaries of the key findings and observations using abstractive summarization techniques such as sequence-to-sequence models or transformer-based models.

- 15 Extractive Summarization: MACQ extracts the most important sentences or phrases from the detailed analyses to create informative summaries using extractive summarization techniques such as TextRank or LexRank.

Quantitative Evidence Integration: MACQ integrates quantitative evidence into the self-critique report to support the key findings and observations.

- 20 Metric Visualization: This component visualizes quantitative metrics using charts, graphs, or tables to provide a clear and intuitive representation of the output quality.

- 25 Statistical Analysis: This component performs statistical analysis on the quantitative metrics to derive insights and support qualitative assessments. Techniques such as hypothesis testing, correlation analysis, or regression analysis are used.

Qualitative Assessment Justification:

Argumentative Structuring: This component structures qualitative assessments using argumentative techniques such as claim-premises-conclusion or thesis-antithesis-synthesis to provide a logical and persuasive justification.

- 30 Evidence-Based Reasoning: This component supports qualitative assessments with evidence from modularized analysis, comparative analysis, contextual evaluation, and limitations identification. Techniques such as evidence retrieval or evidence-based argumentation are used.

MACQ Business Layer

Actors

The key business actors involved in the MACQ component include Data Scientists, AI Engineers, and Domain Experts. Data Scientists are responsible for analysing and
5 interpreting the AI system's outputs, AI Engineers oversee the development and maintenance of the AI system, and Domain Experts provide subject matter expertise for evaluating the quality and accuracy of the outputs.

Services

MACQ offers the business service of Output Quality Assessment and Critique. This service
10 enables a comprehensive and structured evaluation of the AI system's outputs, identifying strengths, weaknesses, and areas for improvement through modularized analysis, quantified metrics, comparative analysis, contextual evaluation, and limitations identification.

Processes

The primary business processes facilitated by MACQ are output evaluation and iterative
15 refinement. This process involves conducting a modularized analysis of the AI system's outputs, quantifying key metrics, comparing the outputs to reference materials and exemplars, evaluating the outputs' contextual alignment, identifying limitations and uncertainties, and synthesizing these findings into a comprehensive critique report. This report serves as a foundation for improving the AI system's performance.

20 Functions

MACQ supports the business functions of Output Quality Assurance, Continuous Improvement, and Knowledge Base Enrichment. It ensures the quality of the AI system's outputs by conducting thorough evaluations and critiques, facilitates continuous
25 improvement by providing actionable insights and recommendations, and enriches the AI system's knowledge base by identifying gaps, biases, and areas for further learning.

Objects

The key business objects involved in the MACQ component are AI Outputs, Reference Materials, Critique Reports, Quantified Metrics, and Identified Limitations. AI Outputs are generated results or responses, Reference Materials are domain-specific knowledge bases
30 and exemplars, Critique Reports encapsulate comprehensive evaluations, Quantified Metrics represent objective measurements of output quality, and Identified Limitations capture

uncertainties, biases, and areas for improvement.

MACQ Application Layer

5 The core application component is MACQ (Modularized Analysis, Quantification, Comparison, and Limitations).

Services

MACQ provides the following application services: Modularized Analysis Service, Metric
Quantification Service, Comparative Analysis Service, Contextual Evaluation Service,
Limitations Identification Service, and Critique Synthesis Service. These services facilitate
10 the modularized analysis of outputs, quantification of metrics, comparison to reference materials, evaluation of contextual alignment, identification of limitations, and synthesis of a comprehensive critique report.

Interfaces

MACQ exposes several application interfaces, including the Output Interface for ingesting
15 AI-generated outputs, the Reference Material Interface for accessing domain-specific knowledge bases and exemplars, the Task Context Interface for obtaining information about the intended audience, purpose, and constraints, and the Reporting and Visualization Interface for presenting the critique report.

Data

20 The application data used within the MACQ component includes AI Outputs, Reference Materials, Task Context Data, Modularized Analysis Data, Quantified Metrics, Comparative Analysis Data, Contextual Evaluation Data, Identified Limitations, and Critique Report. AI Outputs are the generated results or responses; Reference Materials are domain-specific knowledge bases and exemplars; Task Context Data provides information about the
25 intended audience, purpose, and constraints; Modularized Analysis Data represents the structured analysis of outputs across different modules; Quantified Metrics are objective measurements of output quality; Comparative Analysis Data captures the comparison to reference materials; Contextual Evaluation Data assesses the outputs' alignment with the task context; Identified Limitations encapsulate uncertainties and biases; and the Critique
30 Report is the comprehensive synthesis of the evaluation.

Functions

MACQ incorporates the following key application functions:

Modularized Analysis: This function divides the output evaluation into distinct modules, such as relevance, coherence, factual accuracy, completeness, and style. It thoroughly analyzes each module, assessing how well the output meets the defined criteria for that module. §

Metric Quantification: This function generates measurable metrics for each critique module.

- 5 These metrics, such as relevance scores, coherence indices, factual accuracy percentages, completeness ratings, and numerical style ratings, provide an objective assessment of output quality.

- 10 Comparative Analysis: This function compares the AI-generated outputs to relevant reference materials, such as domain-specific knowledge bases, expert-curated examples, or successful outputs from the past. It identifies discrepancies, gaps, or areas for improvement compared to the reference standards. The observed differences are categorized and quantified.

- 15 Contextual Evaluation: This function evaluates the outputs within the broader context of the task. It considers factors like the intended audience, core purpose, and constraints. It assesses whether the outputs align with these contextual factors in terms of content, complexity, tone, and style. It provides an overall contextual fitness score and targeted improvement suggestions.

- 20 Limitations Identification: This function acknowledges any limitations, uncertainties, or potential biases in the AI-generated outputs. It notes unsupported claims, vague language, insufficient elaboration, one-sided argumentation, reliance on heuristics or approximations, and data-driven claims that require statistical qualification. These identified limitations are summarized in a clear and categorized manner.

- 25 Critique Synthesis: This function brings together the findings from the modularized analysis, quantified metrics, comparative analysis, contextual evaluation, and identified limitations. It presents the key findings and observations in a logical flow, using quantitative metrics as evidence to support qualitative assessments. The report includes comparative insights, contextual considerations, and a concise summary of limitations. It concludes with a prioritized list of recommendations and action items to guide iterative improvements.

RAINS Methods

- 30 In the self-critique report, RAINS conducts a root cause analysis for each identified limitation, gap, or area for improvement. This analysis aims to uncover missing knowledge or skills, faulty assumptions, or biases that may have influenced the response. A visual

representation, like a causal graph or a fishbone diagram, is generated to show the relationships between different factors.

Root Cause Analysis:

To identify the underlying factors contributing to the limitations or areas for improvement,

5 RAINS uses a structured root cause analysis. They employ the following techniques:

5 Whys Analysis:

Iterative Questioning: Asking "why" questions iteratively to delve into the root causes of the issues, following the chain of cause-and-effect relationships.

10 Causal Factor Identification: Identifying the causal factors at each level of the 5 Whys analysis, differentiating between necessary and sufficient causes.

Fishbone Diagram Construction:

Cause Categorization: Organizing the potential causes into main categories (e.g., people, process, technology, environment) using a fishbone diagram structure.

15 Cause-Effect Mapping: Mapping the cause-and-effect relationships between the identified causal factors using arrows or lines in the fishbone diagram.

Pareto Analysis:

Frequency Calculation: Determining the frequency or occurrence of each identified causal factor using techniques such as histogram analysis or frequency tables.

20 Cumulative Frequency Computation: Calculating the cumulative frequency or percentage of each causal factor to prioritize the most significant causes using the Pareto principle (80/20 rule).

Causal Graph Generation:

Node Identification: Identifying the nodes representing the issues, causal factors, and their relationships in the causal graph.

25 Edge Determination: Establishing the causal relationships between the nodes using directed arrows or weighted connections.

Bayesian Network Construction:

Variable Selection: Selecting relevant variables for the Bayesian network, including the identified issues, causal factors, and any confounding or mediating variables.

30 Structure Learning: Learning the structure of the Bayesian network from available data using techniques such as constraint-based algorithms (e.g., PC algorithm) or score-based algorithms (e.g., hill climbing).

Parameter Estimation: Estimating the parameters (conditional probabilities) of the Bayesian

network using techniques such as maximum likelihood estimation or Bayesian estimation.

Exploration of Knowledge and Skill Gaps:

- 5 RAINS examine the root cause analysis findings to identify knowledge or skill gaps in the AI system. It specifies the domains, topics, or competencies that need additional knowledge or training.

To explore knowledge and skill gaps, RAINS uses the following techniques:

Domain Knowledge Assessment:

- 10 Knowledge Base Coverage Analysis: Analyzes the AI system's knowledge base to assess the coverage and completeness of knowledge related to identified gaps, using techniques like ontology alignment or knowledge graph comparison.

Domain-Specific Skill Evaluation: Evaluates the AI system's proficiency in domain-specific skills or tasks, using techniques such as performance benchmarking or expert assessment.

Learning Objective Formulation:

- 15 Knowledge Gap Prioritization: Prioritizes knowledge gaps based on their criticality, relevance, and potential impact on the AI system's performance, using techniques like multi-criteria decision analysis or analytic hierarchy process (AHP).

Learning Objective Definition: Defines specific, measurable, achievable, relevant, and time-bound (SMART) learning objectives for each prioritized knowledge gap.

- 20 Training Resource Identification:

Curriculum Design: Designs a structured curriculum or training plan to address the identified knowledge gaps, specifying learning materials, activities, and assessments.

Resource Allocation: Allocates appropriate resources (e.g., time, computational resources, subject matter experts) to support the training and knowledge acquisition process.

- 25 RAINS also analyses the root cause analysis findings to uncover faulty assumptions and biases affecting the AI system's outputs. It determines whether assumptions are explicit or implicit and evaluates their validity against established facts, expert consensus, or diverse perspectives.

For assumption and bias analysis, RAINS uses the following techniques:

- 30 Assumption Identification:

Explicit Assumption Extraction: Extracts explicitly stated assumptions from the AI system's knowledge base, reasoning process, or output, using techniques like rule-based pattern

matching or natural language processing.

Implicit Assumption Inference: Infers implicit assumptions underlying the AI system's reasoning or decision-making, using techniques like logical reasoning or counterfactual analysis.

5 Assumption Validation:

Factual Consistency Checking: Checks the consistency of identified assumptions against facts or knowledge bases, using techniques such as logical inference or knowledge graph querying.

- 10 Expert Consensus Comparison: Compares identified assumptions with expert consensus or domain-specific guidelines, using techniques like the Delphi method or expert elicitation.

Bias Detection:

Cognitive Bias Assessment: This assessment determines whether cognitive biases, such as confirmation bias, anchoring bias, and availability bias, are present in the AI system's reasoning or output by using techniques like heuristic evaluation or cognitive bias checklists.

- 15 Fairness Metric Computation: This computation calculates fairness metrics, such as demographic parity and equalized odds, to identify potential biases in the AI system's predictions or decisions using techniques such as statistical parity difference or disparate impact analysis.

Bias Quantification:

- 20 Bias Impact Estimation: This estimation evaluates the impact or influence of the identified biases on the AI system's outputs through techniques like sensitivity analysis or ablation studies.

Bias Contribution Measurement: This measurement determines the contribution of each identified bias to the overall bias in the AI system's outputs using techniques such as

- 25 Shapley value or feature importance ranking.

- 30 In the process of causal graph refinement, RAINS consolidates the results of root cause analysis, identifies knowledge and skill gaps, and addresses assumptions and biases. This information is then transformed into a clearer representation of the causal graph. The identified limitations are treated as primary nodes, while contributing factors are represented as intermediate nodes. Directed edges are used to establish causal relationships between these nodes.

RAINS uses the following techniques to refine the causal graph:

Node Classification:

Issue Nodes: These nodes represent the identified limitations, gaps, or areas for improvement in the causal graph.

- 5 Factor Nodes: These nodes represent the contributing factors, such as knowledge gaps, skill deficiencies, faulty assumptions, or biases, in the causal graph.

Edge Weighting:

Causal Strength Estimation: This involves estimating the strength or magnitude of the causal relationships between nodes, using techniques like correlation analysis or causal effect estimation.

- 10 Confidence Score Assignment: Confidence scores are assigned to the edges based on the certainty or reliability of the causal relationships. This is done using techniques such as expert judgment or Bayesian updating.

Graph Structure Optimization:

- 15 Cycle Elimination: Any cycles or feedback loops in the causal graph are eliminated, maintaining a directed acyclic graph (DAG) structure. Techniques such as graph traversal or solving the feedback arc set problem are used for this.

Redundancy Removal: Redundant or duplicate nodes and edges are removed from the causal graph to improve clarity and computational efficiency. Techniques like graph simplification or node merging are employed for this.

- 20 Causal Reasoning:

Intervention Simulation: The effect of interventions or changes in the causal graph is simulated using techniques such as do-calculus or structural equation modeling.

- 25 Counterfactual Analysis: The outcomes of counterfactual scenarios or alternative decisions in the causal graph are analyzed. Techniques such as the potential outcomes framework or counterfactual reasoning algorithms are utilized for this.

Insight Synthesis:

RAINS brings together the refined causal graph, analysis of knowledge and skill gaps, assessment of assumptions and biases, and intervention simulations to synthesize actionable insights and recommendations for system improvement. It generates a

comprehensive report summarizing the key findings, prioritized areas for enhancement, and suggested actionable items.

RAINS uses the following techniques for insight synthesis:

Insight Prioritization:

- 5 Impact Assessment: The potential impact of addressing each identified limitation, gap, or bias on the AI system's performance, reliability, and fairness is assessed. Techniques like sensitivity analysis or Pareto optimization are used for this assessment.

- Feasibility Evaluation: Evaluate the feasibility of implementing the proposed interventions or improvements, considering resource constraints, technical complexity, and organizational
10 readiness. Use techniques such as cost-benefit analysis or SWOT analysis.

Recommendation Generation:

Action Item Identification: Identify specific, measurable, and time-bound action items to address prioritized insights. Use techniques like objective and key result (OKR) setting or SMART goal formulation.

- 15 Responsibility Assignment: Assign responsibilities and ownership for each action item to relevant stakeholders or teams. Use techniques like responsibility assignment matrix (RAM) or RACI matrix.

Report Compilation:

- Executive Summary Generation: Generate a concise executive summary highlighting key
20 findings, prioritized insights, and recommended actions. Use techniques like natural language generation or template-based summarization.

- Detailed Analysis Documentation: Document the detailed process of root cause analysis, knowledge and skill gap assessment, assumption and bias evaluation, causal graph refinement, and insight synthesis in a comprehensive report. Use techniques like structured
25 reporting or technical writing.

RAINS Business Layer

The key business actors in the RAINS component are Data Scientists, AI Engineers, and Quality Assurance Specialists. Data Scientists are responsible for conducting root cause analyses and synthesizing insights. AI Engineers oversee the development and

maintenance of the AI system, and Quality Assurance Specialists ensure the robustness and reliability of the root cause analysis process.

Services:

5 RAINS offers the business service of Root Cause Analysis and Insight Synthesis. This service enables structured and comprehensive analysis of the underlying factors contributing to the limitations or areas for improvement identified in the AI system's outputs. It also facilitates synthesizing these insights into actionable recommendations and visual representations for decision-making and system optimization.

Processes:

10 The primary business process facilitated by RAINS is Root Cause Identification and Mitigation. This process involves conducting thorough root-cause analyses to identify the underlying factors contributing to the limitations or gaps in the AI system's outputs, exploring knowledge and skill gaps, analysing faulty assumptions and biases, and synthesizing these findings into causal graphs or other visual representations. These insights inform the
15 development of targeted mitigation strategies and system improvements.

Functions:

RAINS supports several business functions, including Problem Diagnosis, Knowledge Gap Analysis, Bias Mitigation, and Causal Modeling. It helps diagnose underlying problems that are contributing to limitations in output. Additionally, it facilitates the identification and
20 analysis of knowledge and skill gaps within the AI system. It also supports the recognition and mitigation of faulty assumptions and biases. Furthermore, it enables the creation of causal models that represent the relationships between various contributing factors.

The key business objects involved in the RAINS component include Output Limitations, Root Cause Analyses, Knowledge and Skill Gaps, Identified Assumptions and Biases, Causal
25 Graphs, and Mitigation Strategies. Output Limitations represent identified weaknesses or areas for improvement in the AI system's outputs. Root Cause Analyses encapsulate structured analyses of underlying factors. Knowledge and Skill Gaps capture identified gaps in the AI system's knowledge or capabilities. Identified Assumptions and Biases represent faulty assumptions or biases influencing the outputs. Causal Graphs are visual
30 representations of the causal relationships between various factors. Mitigation Strategies outline recommended actions to address the identified root causes.

RAINS Application Layer

The main component of the application is RAINS (Root Cause Analysis and Insight Synthesis).

5 Services

RAINS offers several application services to support root cause analysis and insight synthesis. These services include Root Cause Analysis Service, Knowledge and Skill Gap Exploration Service, Assumption and Bias Analysis Service, Causal Graph Generation Service, and Mitigation Strategy Recommendation Service. These services enable
10 structured root cause analysis, exploration of knowledge and skill gaps, analysis of assumptions and biases, generation of causal graphs, and recommendation of mitigation strategies.

Interfaces

RAINS provides various application interfaces. These interfaces include the Output
15 Limitation Interface, which allows users to input information about identified limitations or areas for improvement. The Knowledge Base Interface provides access to the AI system's knowledge base and skill repositories. The Reporting and Visualization Interface allows the presentation of root cause analyses, causal graphs, and mitigation strategies. The System Adaptation Interface enables integration of recommended mitigation strategies into the AI
20 system's optimization process.

Data

The RAINS component utilizes different types of application data. This includes Output Limitation Data, Root Cause Analysis Data, Knowledge and Skill Gap Data, Assumption and Bias Data, Causal Graph Data, and Mitigation Strategy Data. Output Limitation Data
25 represents identified weaknesses or areas for improvement in the AI system's outputs. Root Cause Analysis Data encompasses structured analyses of underlying factors. Knowledge and Skill Gap Data captures identified gaps in the AI system's knowledge or capabilities. Assumption and Bias Data represents faulty assumptions or biases influencing the outputs. Causal Graph Data provides a structured representation of the causal relationships between
30 various factors. Mitigation Strategy Data outlines recommended actions to address identified root causes.

Functions

RAINS includes the following key application functions:

Root Cause Analysis: This function conducts a structured root cause analysis by applying the 5 Whys methodology or similar techniques. It aims to identify the underlying factors contributing to the identified limitations or areas for improvement. The function categorizes these factors as missing knowledge, inadequate skills, flawed assumptions, biased perspectives, or other relevant types. It also analyzes the interrelationships between these factors.

Knowledge and Skill Gap Exploration: This function reviews the findings of the root cause analysis to identify any knowledge or skill gaps within the AI system. It specifies the domains, topics, or competencies that require additional knowledge or training. It also assesses the importance of each gap based on how it affects the quality of the output. Additionally, it proposes specific learning objectives or training tasks, estimates the time and resources needed, and prioritizes the gaps based on their significance and feasibility.

Assumption and Bias Analysis: This function analyses the findings of the root cause analysis to uncover any faulty assumptions or biases that influence the AI system's outputs. It determines whether these assumptions are explicit or implicit, evaluates their validity against established facts, expert consensus, or diverse perspectives. It categorizes biases as cognitive, social, statistical, or other relevant types, and traces their origins in the training data, architecture, or inference processes. Furthermore, it quantifies the impact of these assumptions or biases on the outputs and provides recommendations for debiasing techniques or mitigation strategies.

Causal Graph Generation: This function integrates the findings of the root cause analysis, knowledge and skill gaps, and identified assumptions and biases into a causal graph representation. It considers the identified limitations as primary nodes and creates intermediate nodes for contributing factors. It establishes the causal relationships between these nodes using directed edges and assigns weights to measure the strength of these connections. The graph is organized hierarchically to show the compounding effects and is validated as a directed acyclic graph (DAG). Finally, the causal graph is serialized into a structured format (e.g., DOT, JSON) for storage and visualization purposes.

Mitigation Strategy Recommendation: Building on the insights gained from the root cause analysis, knowledge and skill gap exploration, assumption and bias analysis, and causal graph representation, this function recommends targeted mitigation strategies to address the identified root causes and limitations. It suggests actions such as updating the knowledge base, modifying the model architecture, employing debiasing techniques, or enriching the

training data. These recommendations are prioritized based on their potential impact and feasibility.

Implementation:

5 To address any limitations, gaps, or areas for improvement identified in the self-critique report, RAINS employs the 5 Whys methodology. This approach involves iteratively asking 'why' questions to dig deeper into the underlying factors. These factors are then categorized as missing knowledge, inadequate skills, flawed assumptions, biased perspectives, or other relevant types. RAINS further analyses the interrelationships between these factors and documents the entire process and findings in a structured format, using the ISO 11179
10 metadata structure. This thorough root cause analysis ensures traceability and reproducibility.

Explore Knowledge and Skill Gaps:

RAINS reviews the root cause analysis to identify any knowledge or skill gaps. Each gap is clearly defined, specifying the domain, topic, or competency for which additional knowledge
15 or training is required. RAINS assesses the criticality of each gap by carefully evaluating its impact on the quality of responses. Based on this assessment, RAINS proposes targeted learning objectives or training tasks, with a focus on the most crucial areas. Additionally, RAINS estimates the time and resources necessary to address each gap and prioritizes them based on their significance and feasibility within the model's architecture.

20 Analyze Assumptions and Biases:

RAINS examines the root cause analysis to uncover any faulty assumptions and biases. It determines whether these assumptions are explicit or implicit in the model's reasoning and evaluates their validity against established facts, expert consensus, or diverse perspectives. To further understand and categorize the biases, RAINS considers them as cognitive, social,
25 statistical, or other relevant types. RAINS also traces the origins of these biases in the training data, architecture, or inference processes. Additionally, RAINS quantifies the influence of each assumption or bias on the response and provides recommendations for debiasing techniques or mitigation strategies.

Generate Causal Graph:

30 To consolidate the root cause analysis, knowledge and skill gaps, and assumptions and biases, RAINS synthesizes these elements into a causal graph. In this graph, the limitations are represented as primary nodes, and intermediate nodes are created for the contributing

factors. RAINS establishes the links between nodes based on their causal relationships, using directed edges. The strength of these causal connections is quantified by assigning edge weights. The causal graph is organized hierarchically, showcasing the compounding effects of the factors, and is validated as a directed acyclic graph (DAG). Finally, RAINS serializes the causal graph into DOT or JSON format for storage and visualization purposes.

IRAP Method

The IRAP (Iterative Response and Process) method involves generating an updated response that addresses identified limitations and incorporates improvements based on self-critique and root cause analysis. This process includes retrieving relevant information, revising the reasoning process, adapting the response's language and style, and incorporating feedback and insights from previous iterations or related tasks. The self-critique and iterative improvement steps are repeated until a satisfactory response or a predefined iteration limit is reached.

Response Generation:

To generate an initial response, IRAP leverages the AI system's natural language generation capabilities, knowledge base, and inference mechanisms. It ensures that the generated response adheres to basic requirements and guidelines.

IRAP employs the following techniques for response generation:

Language Model Adaptation:

Fine-Tuning: Adapting a pre-trained language model (e.g., GPT, BERT) to the specific task or domain using fine-tuning techniques such as transfer learning or domain adaptation.

Prompt Engineering: Designing effective prompts or input sequences to guide the language model towards generating relevant and coherent responses using techniques such as prompt templates or prompt optimization.

Knowledge Base Integration:

Information Retrieval: Retrieving relevant information from the AI system's knowledge base or external sources to inform the response generation using techniques such as semantic search or knowledge graph querying.

Knowledge Grounding: Incorporating relevant facts, entities, or concepts from the retrieved knowledge to ground the generated response using techniques such as entity linking or knowledge-based generation.

Inference and Reasoning:

Logical Inference: Applying logical inference rules or reasoning mechanisms to generate coherent and consistent responses using techniques such as first-order logic or natural language inference.

5 Commonsense Reasoning:

Incorporating commonsense knowledge and reasoning abilities to generate contextually appropriate and plausible responses using techniques such as commonsense knowledge bases or reasoning frameworks.

Style and Tone Adaptation:

10 Linguistic Style Transfer: Adapting the linguistic style and tone of the generated response to match the desired characteristics (e.g., formality, politeness) using techniques such as style transfer or controlled text generation.

Emotion and Sentiment Control: Controlling the emotional tone and sentiment of the generated response to align with the task requirements or user preferences using

15 techniques such as sentiment analysis or emotion-guided generation.

Feedback and Insight Integration: IRAP analyzes data from related tasks and previous iterations to find relevant feedback and insights that can improve the response. It assesses the feedback for its applicability and alignment with the current task's objectives, extracts

20 actionable suggestions, and integrates them into the response generation process.

IRAP uses the following techniques for feedback and insight integration:

Feedback Analysis:

Relevance Assessment: Determines how relevant the feedback or insights are to the current task using techniques like semantic similarity measurement or topic modeling.

25 Actionable Insight Extraction: Identifies actionable suggestions or improvements from the feedback using techniques such as aspect-based sentiment analysis or opinion mining.

Response Adaptation:

Content Enrichment: Adds more information, examples, or explanations to the generated response based on the integrated feedback and insights. This is done using techniques like

30 information fusion or content planning.

Reasoning Refinement: Improves the reasoning process or logic of the generated response using the feedback and insights. Techniques such as argumentation mining or discourse analysis are used for this purpose.

Iterative Improvement:

- 5 Incremental Updating: Updates the generated response incrementally by incorporating the integrated feedback and insights. Text editing or delta encoding techniques are used for this purpose.

- 10 Convergence Assessment: Evaluates the convergence or stability of the iterative improvement process using techniques like similarity measurement or convergence criteria evaluation.

Iterative Improvement Loop:

- 15 IRAP establishes an iterative improvement loop that keeps repeating the self-critique, root cause analysis, and response generation steps until a satisfactory output is achieved or a predefined iteration limit is reached. Each iteration assesses the response quality using the evaluation metrics and criteria defined in the self-critique process.

IRAP uses the following techniques for the iterative improvement loop:

Quality Assessment:

- 20 Metric Computation: Calculates evaluation metrics (e.g., relevance, coherence, factual accuracy, completeness, style) for the generated response using the techniques and criteria defined in the self-critique process.

Threshold Comparison: Compares the computed metrics against predefined quality thresholds or benchmarks to determine if the response meets the desired quality levels.

Iteration Control:

- 25 Convergence Monitoring: Monitors the convergence or improvement of the response quality across iterations using techniques such as delta analysis or trend estimation.

Stopping Criteria: Defines stopping criteria for the iterative improvement loop based on factors such as quality thresholds, iteration limits, or diminishing returns. Techniques like multi-objective optimization or decision analysis are used for this purpose.

Model and Process Adaptation:

Hyperparameter Tuning: Adjusts hyperparameters of underlying models (e.g., language model, knowledge retrieval model) based on iteration outcomes using techniques like grid search or Bayesian optimization.

5 Process Reconfiguration: Reconfigures response generation process or pipeline based on identified bottlenecks or inefficiencies using techniques such as workflow optimization or dynamic process adaptation.

Quality Evaluation:

10 IRAP evaluates generated outputs against predefined criteria and thresholds, including relevance, coherence, factual accuracy, completeness, and stylistic appropriateness. It computes various quality metrics, assigns overall quality scores, and classifies the outputs (e.g., excellent, good, needs improvement, poor, unacceptable).

IRAP employs the following techniques for quality evaluation:

Metric Computation:

15 Relevance Metrics: Computes relevance metrics such as cosine similarity, TF-IDF similarity, or semantic similarity between generated response and input prompt or task description.

Coherence Metrics: Computes coherence metrics such as discourse coherence, topic continuity, or logical consistency within generated response using techniques like coreference resolution or discourse parsing.

20 Factual Accuracy Metrics: Computes factual accuracy metrics by verifying generated response against reliable knowledge sources or fact-checking databases using techniques like named entity recognition or relation extraction.

Completeness Metrics: Computes completeness metrics by assessing coverage of key information or required elements in generated response using techniques like information extraction or completeness scoring.

25 Stylistic Metrics: Computes stylistic metrics such as readability, clarity, or adherence to stylistic guidelines using techniques like readability scoring or style-checking.

Score Aggregation:

30 Weighted Averaging: Aggregates individual quality metrics into an overall quality score using weighted averaging techniques, assigning different weights to each metric based on their importance or relevance.

Multi-Criteria Decision Analysis: Applies multi-criteria decision analysis techniques, such as the Analytic Hierarchy Process (AHP) or Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS), to compute the overall quality score considering multiple criteria and their relative priorities.

5 Output Classification:

Threshold-Based Classification: Classifies generated outputs into predefined categories (e.g., excellent, good, needs improvement, poor, unacceptable) based on overall quality score and predefined threshold values.

Machine Learning Classification: Trains a machine learning classifier (e.g., decision tree, support vector machine) to classify outputs based on their quality metrics and labeled training data.

Metadata Management:

IRAP manages and maintains metadata for the iterative improvement process. This includes iteration logs that document results, decisions, and improvements made in each iteration.

15 The metadata is stored in a structured format, such as JSON, for easy retrieval, analysis, and integration with other components of the AI system.

IRAP implements the following techniques for metadata management:

Metadata Schema Definition:

JSON Schema: Defines the structure and format of the metadata using JSON Schema, specifying required fields, data types, and constraints for each element

ISO 11179 Compliance: Ensures that the metadata schema adheres to the ISO 11179 standard for metadata registries, promoting interoperability and standardization

Metadata Capture and Storage:

25 Automated Logging: Automatically captures relevant metadata (e.g., iteration number, input prompt, generated response, quality metrics, improvement decisions) during each iteration using logging frameworks or instrumentation techniques

Database Integration: Stores captured metadata in a structured database (e.g., relational database, NoSQL database) for efficient querying, retrieval, and analysis

Metadata Analysis and Visualization:

30 Aggregation and Summarization: Aggregates and summarizes metadata across iterations to derive insights and trends using techniques such as data aggregation or statistical summarization

Visual Analytics: Visualizes metadata using charts, graphs, or dashboards to facilitate understanding and interpretation of the iterative improvement process using techniques such as data visualization or visual analytics.

IRAP Business Layer

5 Actors

The key business actors involved in the IRAP component are AI Engineers, Natural Language Processing (NLP) Specialists, and Product Managers. AI Engineers are responsible for developing and maintaining the core functionality of the AI system. NLP Specialists oversee the natural language generation and processing aspects, while Product Managers ensure that the iterative improvements align with product goals and user requirements.

Services

IRAP offers the business service of Iterative Output Improvement and Generation. This service enables the AI system to generate updated and enhanced outputs by incorporating insights from self-critique, root cause analysis, and feedback from previous iterations or related tasks. The outputs are refined until they meet predefined quality thresholds or iteration limits.

Processes: The primary business process facilitated by IRAP is Iterative Response Generation and Refinement. This process involves several steps: generating an initial response to a given prompt or task, conducting a self-critique and root cause analysis to identify areas for improvement, retrieving relevant information and insights from previous iterations or related tasks, and iteratively refining the response by incorporating these inputs until the desired quality level is achieved or a predefined iteration limit is reached.

IRAP supports the business functions of Natural Language Generation, Output Quality Assurance, Knowledge Integration, and Continuous Improvement. It enables the generation of natural language outputs, ensures the quality of these outputs through iterative refinement, facilitates the integration of knowledge and insights from various sources, and drives continuous improvement of the AI system's performance.

The key business objects involved in the IRAP component include Initial Outputs, Self-Critique Reports, Root Cause Analyses, Related Task Data, Improved Outputs, Iteration Logs, and Final Outputs. Initial Outputs represent the AI system's initial response to a given prompt or task, Self-Critique Reports encapsulate the structured evaluations and critiques of

these outputs, Root Cause Analyses capture the underlying factors contributing to identified limitations, Related Task Data provides insights and feedback from previous iterations or related tasks, Improved Outputs are the iteratively refined versions of the initial outputs, Iteration Logs document the improvement process and decisions made, and Final Outputs represent the optimized outputs resulting from the iterative refinement process.

The core application component is IRAP (Iterative Response and Process), which provides several application services: Response Generation Service, Feedback and Insight Integration Service, Iterative Improvement Service, and Quality Evaluation Service. These services facilitate the generation of initial responses, the integration of feedback and insights from various sources, the iterative refinement of responses based on identified areas for improvement, and the evaluation of response quality against predefined criteria, respectively.

IRAP exposes several application interfaces, including the Prompt Interface for receiving input prompts or tasks, the Self-Critique Interface for ingesting self-critique reports and root cause analyses, the Related Task Interface for accessing data from previous iterations or related tasks, the Knowledge Base Interface for retrieving relevant information and insights, and the Reporting and Visualization Interface for presenting the improved outputs, iteration logs, and final outputs.

The IRAP component of the application utilizes various types of data, including Prompt Data, Initial Output Data, Self-Critique Data, Root Cause Analysis Data, Related Task Data, Knowledge Base Data, Improved Output Data, Iteration Log Data, and Final Output Data. Prompt Data represents the input prompts or tasks, Initial Output Data captures the AI system's initial responses, Self-Critique Data and Root Cause Analysis Data record the evaluations and underlying factors, Related Task Data provides insights and feedback from previous iterations or related tasks, Knowledge Base Data represents relevant information accessible to the AI system, Improved Output Data represents refined versions of the initial outputs, Iteration Log Data documents the improvement process and decisions, and Final Output Data represents optimized outputs resulting from the iterative refinement process.

The IRAP component incorporates several key application functions:

1. Response Generation: This function generates an initial response to a prompt or task by utilizing the AI system's natural language generation capabilities, knowledge base, and inference mechanisms. It ensures that the response adheres to basic requirements and guidelines.

2. Feedback and Insight Integration: This function reviews data from related tasks and previous iterations to identify relevant feedback and insights that can further enhance the response. It analyzes the feedback for applicability and alignment with the current task's objectives, extracts actionable suggestions, and integrates them into the response generation process, adapting the language model's behavior and outputs accordingly.

3. Iterative Improvement: This function establishes an iterative improvement loop that repeatedly applies the self-critique, root cause analysis, and response generation steps until a satisfactory output is achieved or a predefined iteration limit is reached. Each iteration assesses the response quality using evaluation metrics and criteria defined in the self-critique process, compares the current iteration's performance with previous iterations to measure improvement, and determines whether the response meets predefined quality thresholds or if further iterations are necessary.

4. Quality Evaluation: This function evaluates the quality of the generated outputs against predefined criteria and thresholds, such as relevance, coherence, factual accuracy, completeness, and stylistic appropriateness. Based on these scores, it quantifies various quality metrics, assigns overall quality scores, and classifies the outputs (e.g., excellent, good, needs improvement, poor, unacceptable).

5. Metadata Management: IRAP is responsible for managing and maintaining metadata associated with the iterative improvement process. This includes the iteration logs, which detail the outcomes, decisions, and enhancements made in each iteration. The metadata is stored in a structured format, such as JSON, to ensure effortless retrieval, analysis, and integration with other components of the AI system.

COGNATE Module

COGNATE (Composable Game-theoretic Nash-embedded Adaptive Techniques for Estimating Confidence) is an advanced system and method that efficiently generates, integrates, and adaptively composes diverse confidence estimation techniques. These techniques are used to produce reliable and well-calibrated confidence scores for AI model outputs. At its core, COGNATE is a synergistic ensemble of specialized methods and strategies that collaborate to analyse, assess, and enhance the reliability and interpretability of confidence estimates through continuous optimization cycles.

Advantages of COGNATE include:

Comprehensive Confidence Estimation Techniques: COGNATE incorporates a wide range of complementary confidence estimation methods. These methods include VETTED

(Verbalized confidence, Token probabilities, prompt Entropy, semantic output clustering, and self-consistency Diagnostics), custom prompting strategies, confidence-based prediction ensembling, and the Hidden State Divergence Metric (HSDM). By incorporating these diverse techniques, COGNATE provides multiple perspectives on model confidence, facilitating a holistic assessment of output reliability.

Game-Theoretic Confidence Integration: COGNATE utilizes a principled game-theoretic approach based on the Nash Embedding Theorem to optimally integrate confidence estimates from multiple techniques. By formulating confidence aggregation as a decentralized mechanism design problem and defining appropriate payoff matrices, COGNATE incentivizes specialization and calibration of individual techniques. This approach ensures an adaptive and robust ensemble composition.

Adaptive Ensembling and Composition: COGNATE employs adaptive ensembling techniques, such as weighted averaging of predictions and threshold voting. These techniques dynamically combine outputs from multiple confidence estimation methods based on their strengths and the specific characteristics of the input data. The composition of techniques is continuously optimized using the computed Nash equilibria, resulting in a compelling and self-adjusting ensemble.

FIG. 3 illustrates a flowchart 300 illustrating operations for managing and integrating multiple confidence estimates in an artificial intelligence system using a game-theoretic approach, according to an embodiment herein.

This flowchart 300 comprises several steps designed to enhance decision-making by leveraging diverse confidence estimation techniques treated as agents in a non-cooperative game.

Obtain Initial Output and Multiple Confidence Estimates 302: The process begins by obtaining an initial output from the AI system, which might involve executing a specific task or query. Simultaneously, multiple confidence estimates are generated for this initial output using various computational techniques. Each of these techniques provides a different perspective or calculation method, contributing to a broader understanding of the confidence level associated with the AI's output.

Define a Non-Cooperative Game Where Each Confidence Estimation Technique is Treated as an Agent 304: This step involves conceptualizing the interactions between different confidence estimation techniques as a non-cooperative game. Each technique, treated as an

independent agent, competes or operates with the goal of maximizing its utility, which in this context is aligned with the accuracy or reliability of its confidence estimate.

Define the Strategy Space for Each Agent 306: For each agent (confidence technique), a strategy space is defined. This space encompasses all possible actions or decisions that the agent can take, typically involving different parameters or thresholds that the technique might adjust to optimize its performance.

Generate a Payoff Matrix That Represents the Incentives for Each Technique 308: A payoff matrix is created next, representing the incentives for each confidence estimation technique based on their performance outcomes. This matrix is crucial for determining how different strategies impact the overall effectiveness of each agent and the system at large.

Compute the Nash Equilibrium of the Game 310: With the payoff matrix established, the next step is to compute the Nash equilibrium of the game. This equilibrium represents a state where no agent has anything to gain by changing their strategy unilaterally, assuming other agents keep their strategies unchanged. Finding the Nash equilibrium helps identify the optimal strategies for each agent that lead to the most stable and efficient outcomes for the system.

Aggregate the Confidence Scores Based on the Nash Equilibrium 312: Finally, the confidence scores generated by each technique are aggregated based on the Nash equilibrium strategies. This aggregation considers the optimal contributions of each technique, leading to a composite confidence score that reflects a balanced, strategic integration of multiple estimates.

The process outlined in FIG. 3 ensures that our AI system not only generates diverse confidence estimates but also intelligently integrates these estimates to produce the most reliable output possible. This game-theoretic approach facilitates an advanced level of decision-making accuracy, crucial for applications in specific fields or applications.

Agents of COGNATE:

Verbalized Confidence (VCA): This function generates verbalized confidence scores and explanations. It applies HSDM adjustment using adaptive projection and context vectors, performs domain knowledge integration, quantifies uncertainty, and provides comprehensive confidence outputs.

Token Probabilities (TPA): TPA estimates confidence based on token generation probabilities. It adjusts using HSDM correction vector, performs calibration through temperature/Platt scaling, and quantifies token-level uncertainty.

- 5 Prompt Entropy (PE): PE estimates confidence based on the entropy of the probability distribution over target concepts. It adjusts using HSDM vector and context scaling, enabling robust entropy-based confidence estimates.

Proofreading:

- 10 Semantic Output Clustering (SOC) estimates confidence by clustering semantically similar output sequences using advanced clustering algorithms and entropy calculations, with HSDM adjustment for context awareness.

Self-Consistency (SC) estimates confidence based on consistency across multiple runs, using metrics such as percentage agreement, prediction entropy, weighted combinations, counterfactual reasoning, and causal inference for robustness.

- 15 Chain-of-Thought Prompting (CoTP) improves confidence estimation through a step-by-step prompted reasoning process, providing explanations before producing the final confidence score, enhancing interpretability.

Self-Evaluation Prompting (SEP) enhances confidence calibration through a two-stage prediction and self-reflection process, adjusting confidence scores based on self-evaluation of correctness.

- 20 Socratic Questioning (SQ) systematically guides confidence estimation through an iterative sequence of focused questions, enabling step-by-step reasoning analysis and context-aware confidence.

Weighted Averaging (WAP) combines predictions and confidence estimates from multiple techniques using a weighted average based on individual confidence scores.

- 25 Threshold Voting (TV) generates ensemble predictions by treating each model as a "vote" weighted by confidence, selecting the output with the maximum weighted vote total.

Nash Embedding (NECI) is a game-theoretic approach to optimally combine confidence estimates with techniques specializing based on their empirical accuracies for different input patterns.

- 30 Evaluation & Validation

Evaluation of COGNATE includes:

Confidence Calibration Metrics: Expected Calibration Error (ECE) and Maximum Calibration Error (MCE) to quantify the alignment between predicted confidence and actual accuracy across techniques.

5 Domain Validation Sets: Curated datasets representing a variety of input patterns, domain complexities, and real-world use cases to assess generalizability.

Comparative Analysis: Extensive benchmarking against state-of-the-art confidence estimation methods and established baselines to identify relative strengths and complementarity.

10 User & Expert Evaluation: A qualitative assessment by domain experts and end-users to validate the usefulness, interpretability, and decision-support value of the confidence outputs.

Ablation Studies: Isolating the individual impact of each confidence estimation technique and COGNATE component through selective ablation experiments.

Design of COGNATE:

15 COGNATE provides a unified system for integrating diverse confidence estimation techniques. COGNATE also supports uncertainty-aware decision-making and planning by providing methods and algorithms for incorporating confidence estimates into the reasoning and decision-making processes. By considering the uncertainty associated with predictions and decisions, COGNATE enables AI reasoners to make more informed and risk-aware
20 choices. This is particularly important in scenarios where the consequences of incorrect or unreliable predictions can be significant. COGNATE includes techniques for optimizing decisions under uncertainty, identifying robust and sensitive decision boundaries, and adapting strategies based on the confidence levels associated with different outcomes. By integrating uncertainty-aware decision-making into AI systems, COGNATE enhances their
25 reliability and effectiveness in real-world applications.

In conclusion, COGNATE is an advanced system that addresses the need for comprehensive and context-aware confidence estimation in AI reasoners. By integrating diverse estimation techniques, incorporating relevant domain knowledge, and supporting adaptive ensemble learning and uncertainty-aware decision-making, COGNATE provides AI
30 systems with the tools and capabilities to generate accurate, reliable, and meaningful confidence estimates. This ultimately enhances the transparency, trustworthiness, and usability of AI systems in various domains and applications.

COGNATE allows AI reasoners to effectively utilize confidence estimates in reasoning and action selection processes. This involves developing techniques for incorporating uncertainty information into decision-making algorithms, such as Bayesian optimization, robust optimization, or stochastic programming. COGNATE provides methods for propagating and updating uncertainty estimates throughout the reasoning chain, enabling AI reasoners to make informed decisions based on the confidence associated with intermediate results and assumptions. Additionally, COGNATE includes mechanisms for generating contingency plans or adaptive strategies that can handle different levels of uncertainty and adjust the AI reasoner's behavior accordingly. By incorporating uncertainty-aware decision-making and planning, the system and methods can enable AI reasoners to operate more effectively in situations with incomplete or noisy information.

COGNATE ensures the long-term reliability and validity of the generated estimates by continuously monitoring and calibrating confidence estimation models. Techniques are employed to track and analyze the performance of these models over time, identifying potential drifts, biases, or miscalibrations. COGNATE includes methods for detecting and diagnosing issues such as overconfidence, under-confidence, or poor discrimination, and provides mechanisms for automatically triggering recalibration or retraining of the models when necessary. Additionally, COGNATE incorporates techniques for actively seeking and incorporating feedback from users or external sources to validate and refine the confidence estimates. By implementing continuous monitoring and calibration capabilities, the system and methods can maintain the integrity and trustworthiness of the confidence estimates, adapting to evolving data landscapes and user requirements.

Business Layer of COGNATE:

The key business actors involved in the COGNATE system and methods include AI Engineers, Data Scientists, and Domain Experts. AI Engineers are responsible for developing and maintaining the AI models and confidence estimation techniques, while Data Scientists analyze and interpret the confidence estimates and model outputs. Domain Experts provide subject matter expertise for evaluating the accuracy and reliability of the confidence estimates.

Services

COGNATE offers the business service of Reliable Confidence Estimation and Integration for AI Model Outputs. This service enables the generation of well-calibrated and robust confidence scores for AI model outputs by integrating diverse confidence estimation techniques using game-theoretic principles and adaptive ensembling strategies.

Processes:

The primary business processes facilitated by COGNATE are confidence estimation and model output refinement. This process involves generating confidence estimates for AI model outputs using various techniques (such as VETTED methods, custom prompting strategies, and hidden state divergence Metrics), integrating these estimates using game-theoretic principles and adaptive ensembling, and refining the model outputs based on the integrated confidence estimates.

Functions

COGNATE supports the business functions of Model Performance Evaluation, Confidence Calibration, Ensemble Learning, and Continuous Improvement. It enables the evaluation of AI model performance by providing reliable confidence estimates, facilitates the calibration of confidence scores to align with actual accuracies, employs ensemble learning techniques to combine multiple confidence estimation methods, and drives continuous improvement of model outputs by leveraging the integrated confidence estimates.

Objects

The key business objects involved in the COGNATE system and methods include AI Model Outputs, Confidence Estimates, Integrated Confidence Scores, Payoff Matrices, Nash Equilibria, and Refined Model Outputs. AI Model Outputs are the generated results or predictions, Confidence Estimates are the individual confidence scores from various techniques, Integrated Confidence Scores are the combined confidence estimates from the game-theoretic integration, Payoff Matrices represent the relative accuracies of techniques for different input patterns, Nash Equilibria are the optimal strategies for composing the techniques, and Refined Model Outputs are the improved outputs based on the integrated confidence estimates.

Application Layer

Component

The core application component is COGNATE (Composable Game-theoretic Nash-embedded Adaptive Techniques for Estimating Confidence).

Services

COGNATE provides the following application services: VETTED Confidence Estimation Service, Custom Prompting Service, Hidden State Divergence Metric (HSDM) Service,

Game-Theoretic Integration Service, Adaptive Ensembling Service, and Model Output Refinement Service. These services facilitate the generation of confidence estimates using various techniques, the incorporation of custom prompting strategies, the calculation of HSDM, the game-theoretic integration of confidence estimates, the adaptive ensembling of multiple techniques, and the refinement of model outputs based on the integrated confidence estimates.

Interfaces:

COGNATE exposes several application interfaces, including the Model Output Interface for ingesting AI model outputs, the Domain Knowledge Interface for accessing domain-specific knowledge bases and taxonomies, the Reporting and Visualization Interface for presenting confidence estimates and refined outputs, and the Model Update Interface for integrating the refined outputs back into the AI model.

Data:

The COGNATE system and methods use various types of application data, including AI Model Outputs, Domain Knowledge Data, Confidence Estimation Data, Payoff Matrix Data, Nash Equilibrium Data, Integrated Confidence Data, and Refined Output Data. AI Model Outputs are generated results or predictions, Domain Knowledge Data represents domain-specific knowledge bases and taxonomies, Confidence Estimation Data includes individual confidence estimates from different techniques, Payoff Matrix Data stores accuracies of techniques for different input patterns, Nash Equilibrium Data represents optimal strategies for composing techniques, Integrated Confidence Data captures combined confidence estimates from game-theoretic integration, and Refined Output Data contains improved outputs based on integrated confidence estimates.

Functions:

COGNATE incorporates the following key application functions:

VETTED Confidence Estimation: This function generates confidence estimates using techniques from the VETTED suite, such as verbalized confidence scores, token probability calculations, prompt entropy measurements, semantic output clustering, and self-consistency evaluations.

Custom Prompting: This function uses advanced prompting strategies, like chain-of-thought prompting, self-evaluating prompts, and Socratic questioning, to elicit more informative and self-aware predictions from AI models.

Hidden State Divergence Metric (HSDM): This function calculates the HSDM by measuring the divergence between hidden state representations corresponding to genuine and hallucinated model responses. This provides an additional layer of confidence estimation by capturing the model's awareness of potential inconsistencies or inaccuracies.

- 5 Game-Theoretic Integration: This function implements the game-theoretic approach based on the Nash Embedding Theorem to optimally integrate confidence estimates from multiple techniques. It constructs payoff matrices representing the relative accuracies of techniques, computes the Nash equilibrium to determine the optimal composition strategy, and combines the individual confidence scores using a weighted aggregation scheme based on the
10 equilibrium proportions.

Adaptive Ensembling: This function employs adaptive ensembling techniques, such as weighted averaging of predictions and threshold voting, to effectively combine outputs from multiple confidence estimation methods. It dynamically adjusts the ensemble based on the confidence scores and the specific characteristics of the input data.

- 15 Model Output Refinement: This function refines the outputs of the AI model based on confidence estimates from game-theoretic integration and adaptive ensembling techniques. It identifies and prioritizes outputs with low confidence scores and implements strategies to improve or correct them, such as additional training, data augmentation, or human-in-the-loop interventions.

- 20 Verbalized Confidence:

Verbalized Confidence (VCA) Agent Methods:

- The VCA agent provides reliable and informative confidence scores for concept mapping between input and target domains. It leverages advanced techniques to generate verbalized confidence scores, which are then adjusted using the Hidden State Divergence Metric
25 (HSDM). Additionally, domain-specific knowledge is incorporated to enhance reliability and interpretability.

Verbalized Confidence Score Generation:

- VCA begins by generating a genuine response that accurately maps the input concept to the most suitable concept in the target domain. It also generates a hallucinated response that
30 provides an inconsistent or incorrect mapping. By comparing the hidden state representations of these responses, VCA calculates the HSDM vector, which indicates the model's awareness of potential inconsistencies or inaccuracies in its predictions.

VCA employs the following techniques for generating verbalized confidence scores:

Genuine Response Generation:

5 Concept Embedding: Embeds the input concept and target domain concepts into a high-dimensional vector space using techniques like word embeddings or contextualized embeddings.

Similarity Computation: Computes the similarity between the input concept embedding and each target domain concept embedding using measures such as cosine similarity, Euclidean distance, or dot product.

10 Response Decoding: Decodes the genuine response by selecting the target domain concept with the highest similarity score and generating a natural language mapping using techniques like template-based generation or sequence-to-sequence models.

Hallucinated Response Generation:

15 Inconsistency Induction: Generates an inconsistent or incorrect mapping by selecting a target domain concept with a low similarity score or by introducing random perturbations to the genuine response using techniques like noise injection or adversarial example generation.

Response Decoding: Decodes the hallucinated response using similar techniques as in the genuine response generation.

20 HSDM Computation:

Hidden State Extraction: Extracts the hidden state representations of the genuine and hallucinated responses from the language model at different layers and time steps using techniques such as model introspection or probing.

25 Divergence Measurement: Measures the divergence between the hidden state representations using distance metrics like Euclidean distance, KL divergence, or Jensen-Shannon divergence.

UQC assesses the uncertainty in the predictions made by the model and calibrates the confidence scores to improve the reliability of the predictions. It employs the following techniques:

30 Uncertainty Estimation:

Bayesian Inference: Uses Bayesian methods to estimate the uncertainty associated with the model's predictions, considering both aleatoric and epistemic uncertainties

Dropout: Applies dropout during model training to approximate the posterior distribution of the model's parameters, allowing for uncertainty estimation

5 Confidence Calibration:

Platt Scaling: Applies Platt scaling to map the confidence scores to calibrated probabilities, ensuring that the confidence scores are well-calibrated and correspond to the true probability of correctness.

10 Isotonic Regression: Uses isotonic regression to further refine the calibration of the confidence scores, particularly in cases where Platt scaling is not sufficient.

By incorporating uncertainty quantification and calibration techniques, UQC enhances the model's ability to provide reliable and well-calibrated confidence scores, leading to more trustworthy predictions. VCA incorporates techniques to quantify uncertainty, improving the interpretability and reliability of adjusted confidence scores. It estimates uncertainty
15 associated with adaptive projection and context-specific scaling processes, which are then included in the final adjusted confidence score. Additionally, VCA includes validation and calibration steps to ensure the effectiveness and accuracy of the adjusted scores.

VCA utilizes the following techniques for uncertainty quantification and calibration:

Uncertainty Estimation:

20 Bootstrapping: Generates multiple instances of adaptive projection and context-specific scaling processes using bootstrapping techniques. Uncertainty is estimated by computing the variance or standard deviation of the resulting scores.

Bayesian Inference: Uses Bayesian inference techniques such as Gaussian processes or Bayesian neural networks to model uncertainty in the adaptive projection and context-
25 specific scaling processes.

Calibration:

Reliability Diagram: Constructs a reliability diagram that compares the adjusted confidence scores with observed correctness probabilities using a calibration dataset.

Isotonic Regression: Calibrates the adjusted confidence scores by applying isotonic
30 regression. It learns a monotonic transformation that minimizes the calibration error.

Platt Scaling: Applies Platt scaling to calibrate the adjusted confidence scores. It learns a logistic regression model that maps the scores to calibrated probabilities.

Domain Knowledge Integration:

5 VCA incorporates domain-specific knowledge to make the HSDM-based correction more relevant and tailored to the specific domain. It utilizes domain-specific ontologies, taxonomies, or semantic networks to guide the adaptive projection and context-specific scaling processes. By incorporating domain knowledge, the HSDM-based correction becomes more aligned with the specific requirements and intricacies of the domain.

VCA employs the following techniques for domain knowledge integration:

10 Ontology Alignment:

Concept Mapping: Maps input and target domain concepts to a domain-specific ontology or taxonomy using techniques like string matching, synonym expansion, or semantic similarity.

Hierarchical Encoding: Encodes hierarchical relationships between concepts in the ontology using techniques like path-based encoding or tree-based convolution.

15 Semantic Embedding:

Knowledge Graph Embedding: Embeds concepts and relationships in a domain-specific knowledge graph into a dense vector space using techniques like TransE, DistMult, or ComplEx.

20 Semantic Similarity Computation: Computes semantic similarity between concepts based on their embeddings and the structure of the knowledge graph using techniques like graph convolution or attention mechanisms.

Domain-Specific Constraint Integration:

25 Constraint Encoding: This process includes encoding domain-specific constraints, such as mutual exclusivity or implication relationships, into the adaptive projection and context-specific scaling processes. This is done using techniques like logical rules or penalty terms.

Constraint Optimization: This process optimizes the adaptive projection and context-specific scaling to meet the domain-specific constraints. Techniques like constrained optimization or integer programming are used for this purpose.

Comprehensive Output:

The output of VCA consists of the predicted mapping between the input and output concepts, along with the HSDM-adjusted confidence score. This adjusted score incorporates uncertainty estimates, the calibrated score, and the domain-adapted correction factor. This comprehensive output provides a more reliable and informative assessment of the model's confidence in its prediction, enabling users to make well-informed decisions based on the reliability of the concept mapping.

VCA generates the comprehensive output using the following techniques:

Output Formatting:

JSON Serialization: The output is serialized into a JSON format that includes the predicted mapping, HSDM-adjusted confidence score, uncertainty estimates, calibrated score, and domain-adapted correction factor.

ISO 11179 Compliance: The output format is designed to adhere to the ISO 11179 standard for metadata registries, thereby promoting interoperability and standardization.

Output Interpretation:

Confidence Score Interpretation: This provides guidelines for interpreting the HSDM-adjusted confidence score. It includes information about the range of values, the meaning of high and low scores, and the implications for decision-making.

Uncertainty Interpretation: This explains the significance of the uncertainty estimates and how they can be used to assess the reliability and robustness of the predicted mapping.

Domain-Specific Interpretation: This section offers insights into how the domain-adapted correction factor impacts the confidence score and its alignment with the specific domain requirements and nuances.

Agent: Token Probabilities

Token Probabilities Agent (TPA) Methods

The TPA agent utilizes token generation probabilities to estimate the confidence of a language model's predictions. It incorporates advanced techniques, including the Hidden State Divergence Metric (HSDM), adaptive projection, context-specific scaling, calibration, and uncertainty quantification. This ensures that the confidence estimates provided are both well-calibrated and context-aware.

Calculation of Token Probabilities:

TPA uses the language model to generate the output token sequence for the given input. It then calculates initial confidence estimates based on the token probabilities associated with the generated sequence.

TPA employs the following techniques for calculating token probabilities:

5 Language Model Inference:

Autoregressive Generation: By conditioning on the input and previously generated tokens, an autoregressive language model (e.g., GPT, BERT) predicts the next token iteratively, thereby generating the output token sequence.

10 Beam Search Decoding: Beam search decoding is applied to explore multiple sequences of tokens with high probabilities. The most likely output is selected based on the cumulative token probabilities.

Token Probability Extraction:

Softmax Probability: The softmax probability associated with each generated token is extracted from the language model's output distribution.

15 Log Probability: To handle small probability values and prevent underflow issues, the log probability of each generated token is computed.

Confidence Score Computation:

Sequence Probability: The overall confidence score for the generated sequence is calculated by taking the product of the token probabilities (or summing the log probabilities).

20 Length Normalization: Techniques like dividing by the sequence length or using a length penalty term are applied to account for varying sequence lengths.

HSDM-based Confidence Adjustment:

25 To adjust the confidence estimates, TPA utilizes the Hidden State Divergence Metric (HSDM). By measuring the divergence between the hidden state representations of genuine and hallucinated responses, TPA obtains the HSDM vector. Adaptive projection is then applied to transform this high-dimensional vector into a lower-dimensional space while preserving its structural information. Lastly, the transformed vector is compared to a context-specific reference vector using cosine similarity. The token probabilities are scaled by multiplying them with a factor derived from the context-specific similarity score.

30 TPA employs the following techniques for HSDM-based confidence adjustment:

HSDM Computation:

Hidden State Extraction: The hidden state representations of genuine and hallucinated responses are extracted from the language model at different layers and time steps.

- 5 Divergence Measurement: The divergence between the hidden state representations is measured using distance metrics such as Euclidean distance, KL divergence, or Jensen Shannon divergence.

HSDM Vector Construction: The HSDM vector is constructed by concatenating the divergence measures at different layers and time steps, and normalizing the vector.

Adaptive Projection:

- 10 Dimensionality Reduction: Dimensionality reduction techniques, such as PCA or t-SNE, are applied to project the high-dimensional HSDM vector into a lower-dimensional space while preserving its structural information.

- 15 Projection Matrix Learning: An adaptive projection matrix is learned to optimize the preservation of critical structural details using techniques such as metric learning or manifold learning.

Context-Specific Scaling:

Reference Vector Generation: A context-specific reference vector is generated by leveraging domain knowledge, semantic similarities, or task-specific information.

- 20 Similarity Computation: The cosine similarity between the transformed HSDM vector and the context-specific reference vector is computed.

Scaling Factor Determination: The scaling factor is determined based on the context-specific similarity score, where higher similarity leads to less scaling of token probabilities.

Calibration Techniques:

- 25 TPA incorporates calibration techniques to align the model's output probabilities with the actual likelihood of correctness. Temperature scaling and Platt scaling are used to calibrate the token probabilities.

TPA employs the following techniques for calibration:

Temperature Scaling:

Temperature Parameter Optimization: The temperature parameter is optimized using a validation set to minimize the negative log-likelihood loss.

Probability Calibration: The token probabilities are calibrated by dividing the logits by the optimized temperature parameter before applying the softmax function.

5 Platt Scaling:

Logistic Regression Training: A logistic regression model is trained on a calibration dataset to learn the mapping between the uncalibrated token probabilities and the true probabilities.

Probability Calibration: The token probabilities are calibrated by passing them through the trained logistic regression model.

10 Uncertainty Quantification:

TPA explores techniques for quantifying uncertainty to provide a more nuanced understanding of the model's confidence. Monte Carlo dropout is used to estimate the uncertainty associated with the token probabilities.

TPA employs the following techniques for uncertainty quantification:

15 Monte Carlo Dropout:

Dropout Inference: Dropout is applied to the language model during inference to generate multiple output sequences for the same input.

Probability Aggregation: The token probabilities across multiple dropout iterations are aggregated by computing the mean and standard deviation.

20 Uncertainty Estimation: The uncertainty of the token probabilities is estimated by considering the variance or entropy of the probabilities across dropout iterations.

Context-Aware Confidence Estimation: TPA combines the HSDM-adjusted token probabilities with calibration techniques and uncertainty quantification to provide context-aware confidence estimates. The final confidence scores represent the model's calibrated and context-specific assessment of its predictions.

25

The following techniques are used by TPA to generate context-aware confidence estimates:

Confidence Score Integration:

Weighted Averaging: This technique combines the HSDM-adjusted token probabilities, calibrated probabilities, and uncertainty estimates using a weighted averaging scheme.

Adaptive Weighting: TPA learns the optimal weights for combining the different components based on their reliability and relevance to the specific context.

Contextual Interpretation:

- 5 Domain Knowledge Integration: TPA incorporates domain-specific knowledge and constraints to interpret the confidence scores.

Task-Specific Thresholds: TPA defines task-specific confidence thresholds based on the requirements and risks associated with the application.

Output Formatting:

- 10 JSON Serialization: TPA serializes the context-aware confidence estimates, along with the token probabilities, HSDM vector, calibration parameters, and uncertainty estimates, into a JSON format.

ISO 11179 Compliance: TPA ensures that the output format adheres to the ISO 11179 standard for metadata registries, promoting interoperability and standardization.

- 15 Agent: Prompt Entropy

Prompt Entropy (PE) Agent Methods

- When presented with various label options, the PE agent assesses the confidence of a language model's predictions by measuring the uncertainty in its output distribution. To achieve this, it utilizes advanced techniques such as entropy calculation, HSDM-based confidence adjustment, adaptive projection, context-specific scaling, and calibration. These techniques ensure that the confidence estimates provided by the PE agent are robust and contextually aware.

Generation of Probability Distribution:

- 25 The PE agent begins by creating a prompt that includes the input concept and lists all potential target concepts as label options. Subsequently, the language model generates a probability distribution that reflects its preference for each label option.

The following techniques are employed by the PE agent for probability distribution generation:

Prompt Construction:

Label Option Enumeration: Exhaustively lists all possible target concepts as label options, based on the provided target taxonomy or domain knowledge.

- 5 Prompt Template: Constructs a prompt template in a standardized format (e.g., "Input concept: [INPUT_CONCEPT], Label options: [LABEL_OPTIONS]"), which incorporates the input concept and the enumerated label options.

Language Model Inference:

Masked Language Modeling: Utilizes masked language modeling (MLM) to predict the probability distribution over the label options. This involves masking the label position in the prompt template.

- 10 Softmax Probability: Extracts the softmax probabilities associated with each label option from the language model's output distribution.

Probability Normalization:

Softmax Normalization: Ensures that the label probabilities are normalized using the softmax function, guaranteeing that they sum up to 1.

- 15 Logarithmic Scaling: Applies logarithmic scaling to the probabilities to address small values and prevent underflow issues.

- Probability of Error (PE) calculates the entropy of the probability distribution over label options to quantify the uncertainty in the model's prediction. A high entropy value indicates that the model is uncertain about the correct label, while a low entropy value suggests
20 confidence in its prediction.

PE employs the following techniques for entropy calculation:

Shannon Entropy:

Probability Logarithm: Computes the logarithm of each label probability to handle small values and ensure numerical stability.

- 25 Entropy Formula: Calculates the Shannon entropy using the formula: $H(X) = -\sum p(x) \log p(x)$, where $p(x)$ is the probability of each label option.

Normalized Entropy:

Maximum Entropy: Computes the maximum possible entropy for the given number of label options, which occurs when all options have equal probabilities.

Entropy Normalization: Normalizes the calculated entropy by dividing it by the maximum entropy, resulting in a value between 0 and 1.

HSDM-based Confidence Adjustment:

PE obtains the Hidden State Divergence Measure (HSDM) vector by measuring the divergence between the hidden state representations of genuine and hallucinated responses. It applies adaptive projection to transform the high-dimensional HSDM vector into a lower-dimensional space while preserving its structural information. The transformed vector is then compared with a context-specific reference vector using cosine similarity. The entropy-based confidence score is scaled based on the context-specific similarity score.

PE employs the following techniques for HSDM-based confidence adjustment:

HSDM Computation:

Hidden State Extraction: Extracts the hidden state representations of genuine and hallucinated responses from the language model at different layers and time steps.

Divergence Measurement: Measures the divergence between the hidden state representations using distance metrics such as Euclidean distance, KL divergence, or Jensen-Shannon divergence.

HSDM Vector Construction: Constructs the HSDM vector by concatenating the divergence measures at different layers and time steps and normalizing the vector.

Adaptive Projection:

Dimensionality Reduction: Applies dimensionality reduction techniques, such as PCA or t-SNE, to project the high-dimensional HSDM vector into a lower-dimensional space while preserving its structural information.

Projection Matrix Learning: Learns an adaptive projection matrix that optimizes the preservation of critical structural details using techniques such as metric learning or manifold learning.

Context-Specific Scaling:

Reference Vector Generation: Generates a context-specific reference vector by leveraging domain knowledge, semantic similarities, or task-specific information.

Similarity Computation: Computes the cosine similarity between the transformed HSDM vector and the context-specific reference vector.

Scaling Factor Determination: The scaling factor is determined based on the context-specific similarity score, with higher similarity resulting in less scaling of the entropy-based confidence score.

Calibration Techniques:

5

To align the entropy-based confidence scores with the actual likelihood of correctness, PE incorporates calibration techniques. Temperature and Platt scaling techniques are used for this purpose.

PE employs the following techniques for calibration:

10 Temperature Scaling:

Temperature Parameter Optimization: The temperature parameter is optimized using a validation set to minimize the calibration loss, such as negative log-likelihood or Brier score.

15 Confidence Score Calibration: The entropy-based confidence scores are calibrated by applying the optimized temperature parameter to the logits before calculating the probabilities.

Platt Scaling:

Logistic Regression Training: A logistic regression model is trained on a calibration dataset to learn the mapping between the uncalibrated confidence scores and the true probabilities.

20 Confidence Score Calibration: The entropy-based confidence scores are calibrated by passing them through the trained logistic regression model.

Context-Aware Confidence Estimation:

To provide context-aware confidence estimates, PE combines the HSDM-adjusted entropy-based confidence scores with the calibration techniques. The final confidence scores reflect the model's calibrated and context-specific assessment of its predictions.

25 PE generates the context-aware confidence estimates using the following techniques:

Confidence Score Integration:

Weighted Averaging: The HSDM-adjusted entropy-based confidence scores and calibrated scores are combined using a weighted averaging scheme.

Adaptive Weighting: The optimal weights for combining the different components are learned based on their reliability and relevance to the specific context.

Contextual Interpretation:

Domain Knowledge Integration: Domain-specific knowledge and constraints are incorporated into the interpretation of the confidence scores.

Task-Specific Thresholds: Task-specific confidence thresholds are defined based on the requirements and risks associated with the application.

Output Formatting:

JSON Serialization: The context-aware confidence estimates, along with the label probabilities, entropy values, HSDM vector, calibration parameters, and other relevant information, are serialized into a JSON format.

ISO 11179 Compliance: Ensures that the output format adheres to the ISO 11179 standard for metadata registries, promoting interoperability and standardization.

Agent: Semantic Output Clustering

Semantic Output Clustering with Domain Knowledge Adjustment (SOCA) Agent Methods

The SOCA agent utilizes advanced techniques, including clustering algorithms, entropy calculation, HSDM-based confidence adjustment, adaptive projection, context-specific scaling, and domain knowledge integration, to estimate the confidence of a language model's predictions. It assesses the coherence and consistency of the model's outputs using semantic clustering algorithms, resulting in robust and context-aware confidence estimates.

Concept Analysis:

SOCA begins by analyzing the input concept and its definition, identifying key characteristics and criteria that define the concept. It also reviews the target taxonomy and its concepts to gain a comprehensive understanding of the target domain.

SOCA utilizes the following techniques for concept analysis:

Input Analysis:

Semantic Parsing: Applies techniques such as dependency parsing or semantic role labeling to extract the key entities, attributes, and relations from the input concept definition.

Feature Extraction: Extracts relevant features and characteristics of the input concept using techniques such as named entity recognition, keyword extraction, or topic modeling.

Target Review:

- 5 Taxonomy Traversal: Traverses the target taxonomy to identify relevant concepts, their hierarchical relationships, and semantic similarities.

Concept Embedding: Generates vector representations of the target concepts using techniques such as word embeddings (e.g., Word2Vec, GloVe) or contextual embeddings (e.g., BERT, Roberta).

- 10 Output Generation and Semantic Embedding:

SOCA generates diverse output sequences that map the input concept to potential target concepts. Techniques such as top-k sampling or nucleus sampling are used for output generation. Additionally, semantic embeddings are generated for these output sequences to capture their semantic meaning and relationships.

- 15 SOCA employs the following techniques for output generation and semantic embedding:

Output Generation:

Top-k Sampling: Samples the top-k most probable output sequences based on the language model's predicted probabilities.

- 20 Nucleus Sampling: Samples output sequences from the nucleus of the probability distribution, which consists of the smallest set of sequences that contain a cumulative probability mass exceeding a threshold.

Semantic Embedding:

- 25 Sentence Embedding: Generates sentence-level embeddings for the output sequences using techniques such as averaging word embeddings, using the CLS token embedding from BERT, or applying pooling operations on contextual embeddings.

Semantic Similarity: Computes the semantic similarity between the output sequence embeddings using metrics such as cosine similarity, Euclidean distance, or dot product.

- 30 Advanced Clustering Algorithms: SOCA applies advanced clustering algorithms to analyze the coherence and consistency of output sequences by using semantic embeddings. It

explores various techniques, including hierarchical clustering, density-based clustering, and graph-based clustering.

SOCA employs the following techniques for advanced clustering:

Hierarchical Clustering:

- 5 Distance Matrix Computation: It computes the pairwise distance matrix between the semantic embeddings using distance metrics like cosine distance or Euclidean distance.
- Linkage Criteria: It applies linkage criteria (such as single linkage, complete linkage, or average linkage) to determine the similarity between clusters.
- Dendrogram Construction: It constructs a dendrogram to represent the hierarchical structure
- 10 of the clusters based on the linkage criteria.
- Cluster Assignment: It assigns the output sequences to clusters by cutting the dendrogram at a specified level or based on a distance threshold.

Density-Based Clustering (DBSCAN):

- Epsilon Neighborhood: It defines the epsilon neighborhood of a point as the set of points
- 15 within a specified distance (epsilon) from it.
- Core Points: It identifies core points as points with a minimum number of other points (minPts) within their epsilon neighborhood.
- Cluster Expansion: It expands clusters by recursively adding points that are density-reachable from core points.
- 20 Noise Points: It classifies points that are not part of any cluster as noise points.

Graph-Based Clustering:

- Similarity Graph Construction: It constructs a similarity graph where nodes represent the output sequences and edges represent the semantic similarity between them.
- Graph Partitioning: It applies graph partitioning algorithms (such as spectral clustering or
- 25 modularity-based methods) to divide the graph into clusters.
- Cluster Evaluation: It evaluates the quality of the clusters using metrics like silhouette score, modularity, or conductance.

Entropy Calculation:

- SOCA calculates the entropy of the cluster assignments to quantify the uncertainty in the
- 30 clustering results. A high entropy value indicates scattered or overlapping clusters, suggesting uncertainty in the model's predictions. On the other hand, a low entropy value suggests tight and distinct clusters, indicating confidence in the predictions.

SOCA employs the following techniques for entropy calculation:

Cluster Probability Distribution:

Cluster Size Computation: It computes the size of each cluster as the number of output sequences assigned to it.

- 5 Cluster Probability Calculation: It calculates the probability of each cluster by dividing its size by the total number of output sequences.

Shannon Entropy:

- 10 Entropy Formula: It calculates the Shannon entropy using the formula: $H(X) = -\sum p(x) \log p(x)$, where $p(x)$ is the probability of each cluster.

Logarithm Base: It uses a logarithm base of 2 to express the entropy in bits.

Normalized Entropy:

Maximum Entropy: This calculation determines the maximum entropy achievable for a given number of clusters. The maximum entropy occurs when all clusters have equal probabilities.

- 15 Entropy Normalization: This process involves dividing the calculated entropy by the maximum entropy, resulting in a normalized value between 0 and 1.

HSDM based Confidence Adjustment:

- 20 SOCA obtains the HSDM vector by measuring the divergence between the hidden state representations of genuine and hallucinated responses. It then applies adaptive projection to transform the high-dimensional HSDM vector into a lower-dimensional space while preserving its structural information. The transformed vector is then compared with a context-specific reference vector using cosine similarity. The entropy-based confidence score, derived from the semantic output clustering, is scaled based on the context-specific similarity score.

- 25 SOCA employs the following techniques for HSDM-based confidence adjustment:

HSDM Computation:

Hidden State Extraction: Extracts the hidden state representations of genuine and hallucinated responses from the language model at different layers and time steps.

- 30 Divergence Measurement: Measures the divergence between the hidden state representations using distance metrics such as Euclidean distance, KL divergence, or Jensen-Shannon divergence.

HSDM Vector Construction: Constructs the HSDM vector by concatenating the divergence measures at different layers and time steps and normalizing the vector.

Adaptive Projection:

Dimensionality Reduction: Applies dimensionality reduction techniques, such as PCA or t-SNE, to project the high-dimensional HSDM vector into a lower-dimensional space while preserving its structural information.

Projection Matrix Learning: Learns an adaptive projection matrix that optimizes the preservation of critical structural details using techniques such as metric learning or manifold learning.

10 Context-Specific Scaling:

Reference Vector Generation: Generates a context-specific reference vector by leveraging domain knowledge, semantic similarities, or task-specific information.

Similarity Computation: Computes the cosine similarity between the transformed HSDM vector and the context-specific reference vector.

15 Scaling Factor Determination: Determines the scaling factor based on the context-specific similarity score, where higher similarity leads to less scaling of the entropy-based confidence score.

SOCA integrates domain-specific knowledge to enhance the meaningfulness and alignment of HSDM-based correction with a specific domain. It utilizes domain-specific ontologies, taxonomies, or semantic networks to guide adaptive projection and context-specific scaling processes.

SOCA employs the following techniques for domain knowledge integration:

Ontology Alignment:

Concept Mapping: Maps the input concept and target concepts to a domain-specific ontology or taxonomy using techniques like string matching, synonym expansion, or semantic similarity.

Hierarchical Encoding: Encodes the hierarchical relationships between concepts in the ontology using techniques like path-based encoding or tree-based convolution.

Semantic Embedding:

Knowledge Graph Embedding: Embeds the concepts and relationships in a domain-specific knowledge graph into a dense vector space using techniques like TransE, DistMult, or ComplEx.

- 5 Semantic Similarity Computation: Computes the semantic similarity between concepts based on their embeddings and the structure of the knowledge graph using techniques like graph convolution or attention mechanisms.

Domain-Specific Constraint Integration:

- 10 Constraint Encoding: Encodes domain-specific constraints, such as mutual exclusivity or implication relationships, into the adaptive projection and context-specific scaling processes using techniques like logical rules or penalty terms.

Constraint Optimization: Optimizes the adaptive projection and context-specific scaling processes to satisfy domain-specific constraints using techniques like constrained optimization or integer programming.

- 15 Comprehensive Output:

- SOCA provides a comprehensive output that includes the predicted mapping between the input and target concepts, the confidence score for semantic clustering, details of the clustering algorithms used, cluster evaluation metrics, and the HSDM-adjusted confidence score. This comprehensive output enables a thorough understanding of semantic
20 coherence, consistency of the model's predictions, and domain-specific adjustments to enhance confidence estimates.

SOCA generates the comprehensive output using the following techniques:

Mapping Result:

- 25 Cluster-based Mapping: Determines the predicted mapping between the input concept and target concepts based on clustering results, assigning the input concept to the cluster with the highest semantic similarity or the most representative target concept.

Mapping Confidence: Computes a confidence score for the predicted mapping based on cluster assignment probabilities and semantic similarity between the input concept and the mapped target concept.

Clustering Details: Clustering Algorithm Specification: Specifies the clustering algorithms used, such as hierarchical clustering, DBSCAN, or graph-based clustering, along with their hyperparameters and configurations.

- 5 Cluster Evaluation Metrics: Reports the evaluation metrics for each clustering algorithm, such as silhouette score, modularity, or conductance, to assess the quality and coherence of the clusters.

HSDM-adjusted Confidence:

HSDM Vector Representation: Includes the HSDM vector, which captures the divergence between genuine and hallucinated responses, as part of the output.

- 10 Adaptive Projection Details: Provides details on the adaptive projection technique used to transform the HSDM vector, including the dimensionality reduction method and the learned projection matrix.

Context-Specific Scaling: Reports the context-specific reference vector, the similarity score, and the scaling factor used to adjust the entropy-based confidence score.

- 15 Output Formatting:

JSON Serialization: Serializes the comprehensive output into a JSON format, ensuring a structured and machine-readable representation.

ISO 11179 Compliance: Ensures that the output format adheres to the ISO 11179 standard for metadata registries, promoting interoperability and standardization.

- 20 Self-Consistency (SC) Agent Methods

The SC agent uses various techniques to estimate the confidence of a language model's predictions. These techniques include evaluating agreement and stability across different variations of input prompts. The goal is to provide a robust measure of confidence that is not easily influenced by slight variations in the prompts.

- 25 Concept Analysis:

SC begins by analysing the input concept and its definition. This analysis involves identifying the key characteristics and criteria that define the concept. SC also reviews the target taxonomy and its concepts in order to gain a comprehensive understanding of the target domain.

- 30 SC employs the following techniques for concept analysis:

Input Analysis:

Semantic Parsing: This technique involves using methods like dependency parsing or semantic role labeling to extract the key entities, attributes, and relations from the input concept definition.

- 5 Feature Extraction: Relevant features and characteristics of the input concept are extracted using techniques such as named entity recognition, keyword extraction, or topic modeling.

Target Review:

Taxonomy Traversal: SC traverses the target taxonomy to identify relevant concepts, their hierarchical relationships, and semantic similarities.

- 10 Concept Embedding: Vector representations of the target concepts are generated using techniques such as word embeddings (e.g., Word2Vec, GloVe) or contextual embeddings (e.g., BERT, RoBERTa).

Prompt Variation Generation:

- 15 SC generates multiple prompt variations that rephrase or paraphrase the original instruction in order to map the input concept to potential target concepts. These variations capture different ways of expressing the same mapping task while preserving the core meaning and intent.

SC employs the following techniques for prompt variation generation:

Paraphrasing:

- 20 Rule-based Paraphrasing: This technique involves applying rule-based methods, such as synonym substitution, active-passive voice conversion, or clause reordering, to generate paraphrased versions of the original prompt.

- Neural Paraphrasing: Neural paraphrasing models, such as sequence-to-sequence models or transformer-based models, are used to generate semantically equivalent prompt variations.
- 25

Prompt Templates:

- Template Construction: SC constructs a set of prompt templates that capture different ways of expressing the mapping task. For example, "What is the most appropriate concept in [TARGET_TAXONOMY] for [INPUT_CONCEPT]?" or "Given [INPUT_CONCEPT], which concept in [TARGET_TAXONOMY] best represents it?"
- 30

Template Filling: The prompt templates are filled with specific input concept and target taxonomy information in order to generate concrete prompt variations.

Prediction Consistency Assessment: SC combines the results from the prediction consistency assessment and entropy calculation using a weighted approach. The weights for each assessment metric can be adjusted based on its importance and the specific requirements of the task.

By assigning appropriate weights to the Fleiss' kappa score, percentage agreement, and normalized entropy, SC obtains a weighted combination score. This score provides a comprehensive evaluation of both the consistency of the model's predictions and the level of uncertainty in those predictions.

The weighted combination score allows for a more nuanced analysis of the language model's performance, considering both the accuracy and reliability of its predictions. It helps in making informed decisions and assessing the quality of the model's outputs.

Overall, SC utilizes a combination of these techniques to evaluate and quantify the consistency, uncertainty, and overall performance of the language model in generating predicted mappings for different prompt variations. SC combines the percentage agreement (Fleiss' kappa) and the inverse of the prediction entropy using a weighted combination to obtain the final self-consistency confidence estimate. The weights can be adjusted based on the relative importance of agreement and certainty in the specific context or domain.

The following techniques are employed by SC for weighted combination:

Weight Assignment:

Expert Judgment: Assigns weights to the percentage agreement and inverse entropy based on specialist knowledge and understanding of the domain and task requirements

Empirical Tuning: Tunes the weights empirically using a validation set to optimize the performance and reliability of the self-consistency confidence estimate

Weighted Sum:

Normalized Scores: Normalizes the percentage agreement and inverse entropy scores to a common scale (e.g., between 0 and 1) to ensure compatibility.

Weighted Sum Calculation: Calculates the weighted sum of the normalized scores using the assigned weights

Confidence Score Scaling:

Range Adjustment: Adjusts the weighted sum to a desired range (e.g., between 0 and 1) using linear scaling or other transformation techniques

- 5 Threshold-based Binning: Bins the adjusted weighted sum into discrete confidence categories (e.g., low, medium, high) based on predefined thresholds

Counterfactual Reasoning and Causal Inference:

- SC explores counterfactual reasoning and causal inference methods to enhance the robustness of the self-consistency measure. These techniques help reduce the model's sensitivity to specific aspects of the input prompt by introducing counterfactual scenarios or analysing the causal relationships between input features and model predictions.
- 10

The following techniques are employed by SC for counterfactual reasoning and causal inference:

Counterfactual Generation:

- 15 Perturbation-based Counterfactuals: Generates counterfactual prompts by perturbing specific aspects of the input prompt, such as word order, synonyms, or grammatical structure, while preserving the core meaning

Model-based Counterfactuals: Utilizes generative models, such as variational autoencoders or generative adversarial networks, to generate plausible counterfactual prompts

- 20 Causal Effect Estimation:

Causal Graph Construction: Constructs a causal graph representing the relationships between input features, prompt variations, and model predictions

Intervention Simulation: Simulates interventions on specific input features or prompt variations to estimate their causal effect on the model's predictions

- 25 Counterfactual Reasoning: Reasons about the model's behavior under counterfactual scenarios by comparing the predicted outcomes with the observed outcomes

Robustness Evaluation:

- Sensitivity Analysis: This analysis measures the sensitivity of the self-consistency measure to specific input features or prompt variations. It does so by evaluating the change in confidence scores under different counterfactual scenarios.
- 30

Robustness Metrics: These metrics, such as average treatment effect (ATE) or effect size, quantify the model's sensitivity to input variations.

Context-Aware Confidence Estimation:

- 5 The context-aware confidence estimate provided by SC incorporates the self-consistency score, which is a weighted combination of percentage agreement and inverse prediction entropy. This score reflects the model's consistency and stability in its predictions across different prompt variations, considering both agreement and certainty.

SC utilizes the following techniques to generate the context-aware confidence estimate:

Confidence Score Interpretation:

- 10 Score Thresholding: This technique compares the self-consistency score against predefined thresholds to determine the level of confidence (e.g., low, medium, high).

Confidence Interval Estimation: It estimates confidence intervals around the self-consistency score to quantify the uncertainty in the confidence estimate.

Contextual Calibration:

- 15 Domain-specific Calibration: This calibration process adjusts the self-consistency score based on domain-specific knowledge and requirements, modifying the thresholds or confidence intervals accordingly.

- Task-specific Calibration: This calibration process adapts based on the specific task and associated risks and consequences, considering factors like the cost of errors or the
20 importance of precision.

Output Formatting:

JSON Serialization: The context-aware confidence estimate, along with the self-consistency score, percentage agreement, prediction entropy, and other relevant information, is serialized into a JSON format.

- 25 ISO 11179 Compliance: The output format adheres to the ISO 11179 standard for metadata registries, promoting interoperability and standardization.

Agent: Chain-of-Thought Prompting

Chain-of-Thought Prompting (CoTP) Agent Methods

The CoTP agent improves the coherence and reliability of a language model's reasoning process. It does so by prompting the model to explain its thought process step-by-step before providing the final answer and confidence score. This approach induces more coherent multi-step reasoning in language models, grounding the confidence generation process in a logical chain of thoughts.

Input Analysis and Concept Identification:

CoTP begins by prompting the model to analyse the input concept and its definition. This step involves identifying the key characteristics and criteria that define the concept.

CoTP utilizes the following techniques for input analysis and concept identification:

Semantic Parsing: Dependency Parsing: Dependency parsing is applied to extract the syntactic structure of the input concept definition, allowing for the identification of relationships between words and phrases.

Semantic Role Labelling: Semantic role labelling is performed to identify the semantic arguments and their roles within the input concept definition. This includes identifying the agent, patient, or instrument.

Named Entity Recognition:

Entity Extraction: Named entities are extracted from the input concept definition using various techniques, including rule-based matching, machine learning models like Conditional Random Fields or Recurrent Neural Networks, and pre-trained named entity recognizers.

Entity Linking: The extracted entities are linked to their corresponding entries in a knowledge base or ontology using techniques such as string matching, semantic similarity, or graph-based methods.

Keyword Extraction:

Statistical Approaches: Statistical methods, such as TF-IDF (Term Frequency-Inverse Document Frequency) or PMI (Pointwise Mutual Information), are applied to identify the most informative and relevant keywords within the input concept definition.

Graph-based Approaches: A graph representation of the input concept definition is constructed, with nodes representing words and edges representing co-occurrence or semantic relationships. Centrality measures, such as PageRank or TextRank, are then applied to identify the most important keywords.

Target Taxonomy Review:

The CoTP prompts the model to review the target taxonomy and its concepts in order to fully understand the options for mapping the input concept.

- 5 CoTP employs the following techniques for target taxonomy review:

Taxonomy Traversal:

Depth-First Search (DFS): A depth-first search is performed on the target taxonomy graph, starting from the root concept and exploring each branch to its leaf concepts.

- 10 Breadth-First Search (BFS): A breadth-first search is performed on the target taxonomy graph, starting from the root concept and exploring all concepts at each level before moving to the next level.

Concept Similarity:

- 15 Semantic Similarity: The semantic similarity between the input concept and each target concept is calculated using techniques such as cosine similarity, Jaccard similarity, or semantic embeddings like Word2Vec or GloVe.

Hierarchical Distance: The hierarchical distance between the input concept and each target concept in the taxonomy is measured using techniques such as the shortest path length or the Wu-Palmer similarity.

Concept Embedding:

- 20 Word Embeddings: Each concept is represented as a dense vector using pre-trained word embeddings like Word2Vec, GloVe, or FastText.

Contextualized Embeddings: Contextualized embeddings are generated for each concept using pre-trained language models like BERT, RoBERTa, or ELMo. These embeddings capture the semantic information within the context of the taxonomy.

- 25 Step-by-Step Reasoning Process: CoTP prompts the model to provide its thoughts on the step-by-step reasoning process, explaining how it arrived at the mapping decision. The model walks through its logical chain of thoughts, considering the similarities and criteria between the input and target concepts.

- 30 CoTP employs the following techniques for the step-by-step reasoning process:

Similarity-based Reasoning:

Semantic Similarity: Calculates the semantic similarity between the input concept and each target concept using the concept embeddings generated in the target taxonomy review step.

- 5 Attribute Matching: Matches the attributes and characteristics of the input concept with those of the target concepts, considering the semantic similarity and relevance of the attributes.

Criteria-based Reasoning:

Rule-based Reasoning: Applies predefined rules or heuristics to evaluate the criteria and constraints for mapping the input concept to the target concepts.

- 10 Constraint Satisfaction: Formulates the mapping problem as a constraint satisfaction problem (CSP) and solves it using techniques such as backtracking, forward checking, or constraint propagation.

Analogical Reasoning:

Structure Mapping: Identifies analogical mappings between the input concept and the target concepts based on their relational structure and similarities.

- 15 Analogical Inference: Infers new properties or relationships for the input concept based on the analogical mappings and the knowledge associated with the target concepts.

Uncertainty Estimation:

Confidence Scores: Assigns confidence scores to each reasoning step based on the strength of the similarity, criteria, or analogical evidence supporting the mapping decision.

- 20 Uncertainty Propagation: Propagates the uncertainty through the reasoning chain using techniques such as Bayesian networks, Dempster-Shafer theory, or fuzzy logic.

Final Answer and Confidence Score:

CoTP prompts the model to provide its final answer for the predicted mapping and a confidence score between 0 and 100 to indicate its level of certainty in the mapping.

- 25 CoTP employs the following techniques for the final answer and confidence score:

Answer Selection:

Weighted Scoring: Assigns weights to each reasoning step based on its importance and combines the confidence scores using a weighted sum or average.

- 30 Decision Thresholding: Compares the combined confidence score against a predefined threshold to determine the final mapping decision.

Confidence Calibration:

Platt Scaling: Applies Platt scaling to calibrate the confidence scores by learning a logistic regression model that maps the scores to probabilities.

5 Isotonic Regression: Uses isotonic regression to calibrate the confidence scores by learning a non-parametric monotonic function that maps the scores to probabilities.

Uncertainty Quantification: Monte Carlo Dropout: Performs multiple forward passes through the model with dropout enabled to estimate the uncertainty in the final answer and confidence score.

10 Bayesian Inference: Applies Bayesian inference techniques, such as variational inference or Markov chain Monte Carlo (MCMC), to estimate the posterior distribution over the mapping decisions and confidence scores.

Coherent Multi-Step Reasoning:

15 CoTP induces more coherent multi-step reasoning in language models by explicitly prompting the model to walk through its reasoning process before providing a final prediction and confidence score. The step-by-step explanation of the model's thoughts helps ensure that the confidence generation process is grounded in a logical chain of reasoning.

CoTP employs the following techniques for coherent multi-step reasoning:

Thought Flow Modeling:

20 Sequence-to-Sequence Models: Uses sequence-to-sequence models, such as recurrent neural networks (RNNs) or transformers, to generate the step-by-step reasoning process as a coherent sequence of thoughts.

Attention Mechanisms: Applies attention mechanisms, such as self-attention or cross-attention, to model the dependencies and relationships between the reasoning steps.

Logical Consistency Checking:

25 Natural Language Inference: Employs natural language inference techniques, such as textual entailment or contradiction detection, to verify the logical consistency between the reasoning steps.

Commonsense Reasoning: Incorporates commonsense knowledge and reasoning capabilities to ensure that the generated reasoning steps are plausible and coherent.

30 Explanation Evaluation:

Human Evaluation: Conducts human evaluation to assess the quality, coherence, and persuasiveness of the generated reasoning explanations.

Automatic Evaluation Metrics: Calculates automatic evaluation metrics, such as BLEU, ROUGE, or BERTScore, to measure the similarity between the generated explanations and reference explanations.

Interpretability and Transparency:

CoTP enhances the interpretability and transparency of the model's reasoning process by requiring it to articulate its thoughts and decision-making steps. This approach provides insights into the model's logic and the factors influencing its mapping decision.

CoTP employs the following techniques for interpretability and transparency:

Explanation Generation:

Template-based Generation: Uses predefined templates or patterns to generate human-readable explanations of the reasoning steps and the factors considered in the mapping decision.

Natural Language Generation: Applies natural language generation techniques, such as rule-based systems or neural language models, to produce fluent and coherent explanations.

Visual Explanation:

Concept Visualization: This component generates visual representations of the input concept, target concepts, and the relationships between them. Techniques such as concept maps, knowledge graphs, or t-SNE plots are utilized for visualization.

Attention Visualization: This component visualizes the attention weights or heatmaps to highlight the essential words, phrases, or concepts that influence the reasoning process.

User Interaction:

Interactive Explanation: This feature provides an interactive interface that allows users to explore the reasoning process, ask questions, and receive clarifications or additional details.

User Feedback Integration: This feature incorporates user feedback and explanations to refine the reasoning process and improve the interpretability and transparency of the model's decisions.

Context-Aware Confidence Estimation:

- 5 The final output of CoTP includes the reasoning process, the final answer, and the confidence score. This comprehensive output provides a context-aware confidence estimation that considers the model's step-by-step thought process, the alignment between the input concept and the target taxonomy, and the model's self-assessment of its certainty.

CoTP generates the context-aware confidence estimation using the following techniques:

Comprehensive Output Formatting:

- 10 JSON Serialization: This technique serializes the reasoning process, final answer, and confidence score into a structured JSON format for easy parsing and interpretation.

Natural Language Summarization: This technique generates a natural language summary of the key points in the reasoning process, highlighting the primary factors and decisions that led to the final answer.

- 15 Contextual Explanation:

Domain Knowledge Integration: This feature incorporates domain-specific knowledge and terminology into the explanations to provide a more contextualized and meaningful interpretation of the reasoning process.

- 20 Comparative Analysis: This feature compares the generated reasoning process and confidence estimation with reference examples or expert explanations to assess their quality and relevance.

Confidence Score Interpretation:

- 25 Uncertainty Quantification: This feature provides quantitative measures of uncertainty, such as confidence intervals or probability distributions, to convey the level of certainty associated with the final answer and confidence score.

Confidence Calibration: This process calibrates the confidence scores using techniques such as Platt scaling or isotonic regression to ensure that they are well-calibrated and aligned with the actual accuracy of the model's predictions.

Agent: Self-Evaluation Prompting:

- 30 Self-Evaluating Prompting (SEP) Agent Methods

The SEP agent improves the calibration of a language model's confidence estimates by using a two-stage process of prediction and self-reflection. It encourages the model to generate an initial prediction and then evaluate the correctness of its answer, providing a confidence score and explanation. This approach enhances the model's ability to assess its performance and provide more accurate confidence estimates.

Concept Analysis:

SEP starts by prompting the model to analyse the input concept and its definition, identifying the key characteristics and criteria that define the concept.

SEP employs the following techniques for concept analysis:

10 Semantic Parsing:

Dependency Parsing: This technique applies dependency parsing to extract the syntactic structure of the input concept definition, identifying the relationships between words and phrases.

15 Semantic Role Labeling: This technique performs semantic role labeling to identify the semantic arguments and their roles in the input concept definition, such as the agent, patient, or instrument.

Named Entity Recognition:

20 Entity Extraction: This technique extracts named entities from the input concept definition using techniques such as rule-based matching, machine learning models (e.g., Conditional Random Fields, Recurrent Neural Networks), or pre-trained named entity recognizers.

Entity Linking: This technique links the extracted entities to their corresponding entries in a knowledge base or ontology using techniques such as string matching, semantic similarity, or graph-based methods.

Keyword Extraction:

25 Statistical Approaches: This technique applies statistical methods, such as TF-IDF (Term Frequency-Inverse Document Frequency) or PMI (Pointwise Mutual Information), to identify the most informative and relevant keywords in the input concept definition.

Graph-based Approaches: This technique constructs a graph representation of the input concept definition, where nodes represent words and edges represent their co-occurrence or

semantic relationships. It then applies centrality measures (e.g., PageRank, TextRank) to identify the most important keywords.

Target Taxonomy Review:

- 5 SEP prompts the model to review the target taxonomy and its concepts to gain a comprehensive understanding of the options for mapping the input concept.

SEP employs the following techniques for target taxonomy review:

Taxonomy Traversal:

- 10 Depth-First Search (DFS): This technique performs a depth-first search on the target taxonomy graph, starting from the root concept and exploring each branch to its leaf concepts.

Breadth-First Search (BFS): This technique performs a breadth-first search on the target taxonomy graph, starting from the root concept and exploring all concepts at each level before moving to the next level.

- 15 Concept Similarity:

Semantic Similarity: Calculates the semantic similarity between the input concept and each target concept using techniques such as cosine similarity, Jaccard similarity, or semantic embeddings (e.g., Word2Vec, GloVe).

- 20 Hierarchical Distance: Measures the hierarchical distance between the input concept and each target concept in the taxonomy using techniques such as the shortest path length or the Wu-Palmer similarity.

Concept Embedding:

- 25 Word Embeddings: Represents each concept as a dense vector using pre-trained word embeddings such as Word2Vec, GloVe, or FastText.

Contextualized Embeddings: Generates contextualized embeddings for each concept using pre-trained language models such as BERT, RoBERTa, or ELMo, capturing the semantic information in the context of the taxonomy.

- 30 Initial Prediction Generation:

SEP prompts the model to generate an initial prediction for mapping the input concept to the most appropriate target concept based on the concept analysis and target taxonomy review.

SEP employs the following techniques for initial prediction generation:

Similarity-based Mapping:

- 5 Cosine Similarity: Calculates the cosine similarity between the input concept embedding and each target concept embedding, selecting the target concept with the highest similarity as the initial prediction.

- 10 Euclidean Distance: Measures the Euclidean distance between the input concept embedding and each target concept embedding, selecting the target concept with the smallest distance as the initial prediction.

Probabilistic Mapping:

Softmax Distribution: Applies a softmax function to the similarity scores or distances to obtain a probability distribution over the target concepts, selecting the target concept with the highest probability as the initial prediction.

- 15 Beam Search: Performs beam search to generate multiple candidate mappings, keeping track of the top-k most probable mappings at each step and selecting the best mapping as the initial prediction.

Rule-based Mapping:

- 20 Heuristic Rules: Defines a set of heuristic rules based on the characteristics and criteria of the input concept and target concepts, applying these rules to determine the most appropriate mapping.

Decision Trees: Constructs a decision tree based on the features and attributes of the input concept and target concepts, traversing the tree to arrive at the initial prediction.

Self-Evaluation and Confidence Scoring:

- 25 SEP prompts the model to self-evaluate the correctness of its initial prediction on a scale of 0-100, assigning a confidence score to indicate its level of certainty in the mapping.

SEP employs the following techniques for self-evaluation and confidence scoring:

Feature-based Evaluation:

Alignment Scoring: Assesses the alignment between the features and characteristics of the input concept and the predicted target concept, assigning higher scores for better alignment.

Criteria Satisfaction: This evaluates how well the predicted target concept meets the criteria and requirements specified in the input concept definition. Higher scores are assigned for greater satisfaction.

Uncertainty Estimation:

Probability Calibration: This technique adjusts the initial prediction probabilities using methods like Platt scaling or isotonic regression. The goal is to obtain well-calibrated confidence scores.

Bayesian Inference: Bayesian inference techniques, such as Bayesian neural networks or Gaussian processes, are applied to estimate the uncertainty in the initial prediction and derive confidence scores.

Comparative Evaluation:

Pairwise Comparison: This involves comparing the predicted target concept with other candidate concepts in the taxonomy. The purpose is to assess their relative suitability and assign confidence scores based on pairwise preferences.

Ranking-based Scoring: Candidate target concepts are ranked based on their similarity, alignment, or criteria satisfaction. Confidence scores are then assigned based on their positions in the ranking.

Explanation of Self-Evaluation:

SEP prompts the model to explain the reasoning behind its self-evaluation score. It highlights the factors contributing to the model's confidence or uncertainty in the initial prediction.

SEP employs the following techniques for generating explanations:

Feature Attribution:

Gradient-based Attribution: This method uses gradient-based approaches, such as saliency maps or integrated gradients, to identify the input features that have the greatest influence on the self-evaluation score.

Perturbation-based Attribution: The input features are perturbed, and the impact on the self-evaluation score is observed. This helps determine the importance of each feature.

Counterfactual Reasoning:

5 Counterfactual Examples: Counterfactual examples are generated by modifying certain features or attributes of the input concept and predicted target concept. This allows for an analysis of the effect on the self-evaluation score.

Counterfactual Explanations: Explanations are provided in the form of counterfactual statements. These statements highlight how changes in specific features or criteria would impact the confidence in the prediction.

10 Natural Language Generation:

Template-based Generation: Predefined templates or patterns are used to generate explanations that are easy for humans to read. These explanations describe the factors contributing to the self-evaluation score.

15 Sequence-to-Sequence Models: Sequence-to-sequence models, such as encoder-decoder architectures or transformers, are employed to generate fluent and coherent explanations. These explanations are based on the input concept, predicted target concept, and self-evaluation score.

Improved Calibration through Self-Reflection:

20 SEP enhances the calibration of the model's confidence estimates by promoting a self-reflective process. This process encourages the model to critically assess its initial prediction and consider potential uncertainties or limitations.

SEP employs the following techniques to improve calibration through self-reflection:

Uncertainty Calibration:

25 Bayesian Calibration: Updates confidence estimates using Bayesian calibration techniques like Bayesian model averaging or Bayesian active learning, based on the model's self-evaluation and uncertainty assessment.

Ensemble Calibration: Improves overall calibration by combining predictions and confidence estimates from multiple models or iterations using ensemble methods such as Bayesian model combination or ensemble Bayesian active learning.

30 Active Learning:

Uncertainty Sampling: Selects the most informative or uncertain examples, based on the

model's self-evaluation scores and confidence estimates, for further evaluation or human feedback.

Query-by-Committee: Utilizes a committee of models to identify examples where the models disagree or have high uncertainty. These examples help refine calibration and improve the self-evaluation process.

Metacognitive Monitoring:

Confidence Thresholding: Sets threshold values for confidence scores to determine when the model should defer to human judgment or seek additional information.

Uncertainty Awareness: Encourages the model to explicitly recognize and communicate its level of uncertainty in the self-evaluation and confidence estimation process.

Interpretability and Transparency:

SEP enhances the interpretability and transparency of the model's confidence estimates by providing explanations of the self-evaluation process and the factors influencing its confidence.

SEP employs the following techniques for interpretability and transparency:

Feature Importance:

Shapley Values: Uses Shapley values from cooperative game theory to assign importance scores to each input feature, indicating its contribution to the self-evaluation score.

Permutation Feature Importance: Measures the importance of each feature by permuting its values and observing the impact on the self-evaluation score.

Concept Activation Vectors:

Concept Embedding Alignment: Aligns the input concept and predicted target concept embeddings with human-interpretable concepts in a pre-defined concept space.

Concept Importance Scoring: Calculates the importance scores for each activated concept based on its contribution to the self-evaluation score and confidence estimate.

Uncertainty Visualization:

Confidence Calibration Plots: Generates plots that compare the model's confidence estimates with the actual accuracy or correctness of its predictions, visualizing the calibration of the confidence scores.

Uncertainty Heat Maps: Creates heat maps or color-coded visualizations that highlight the regions or aspects of the input concept or target taxonomy where the model has high uncertainty or low confidence.

Context-Aware Confidence Estimation:

The final output of SEP includes the initial prediction, the self-evaluation score, and an explanation of the self-evaluation. This comprehensive output provides a context-aware confidence estimation that considers the model's initial prediction, its self-assessment of correctness, and the reasoning behind its confidence score.

SEP generates the context-aware confidence estimation using the following techniques:

Comprehensive Output Formatting:

JSON Serialization: Serializes the initial prediction, self-evaluation score, and explanation into a structured JSON format for easy parsing and interpretation.

Metadata Annotation: Annotates the output with metadata that captures the context, assumptions, and limitations of the confidence estimation process.

Contextual Explanation:

Domain Knowledge Integration: Incorporates domain-specific knowledge and terminology into the explanations to provide a more contextualized and meaningful interpretation of the confidence estimates.

Comparative Analysis: Compares the generated confidence estimates with baseline models or human expert judgments to assess their relative accuracy and reliability.

Confidence Score Interpretation:

Uncertainty Quantification: Provides quantitative measures of uncertainty, such as confidence intervals or probability distributions, to convey the level of certainty associated with the confidence estimates.

Calibration Assessment: Evaluates the calibration of the confidence scores using metrics such as Brier score, log loss, or expected calibration error (ECE), providing insights into the reliability of the confidence estimates.

Agent: Socratic Questioning:

Socratic Questioning (SQ) Agent Methods

The SQ agent uses a series of focused questions to estimate the confidence of a language model's predictions. This iterative process guides the model towards a final prediction and provides a confidence estimate. By employing a structured approach, the SQ agent is able

to systematically guide the model's reasoning process in a more controlled manner. This allows for a step-by-step analysis of the input concept and its mapping to the target concept.

Concept Categorization:

5 SQ starts by asking the model to identify the domain or area of the input concept, providing a broad categorization that serves as a foundation for further analysis.

SQ employs the following techniques for concept categorization:

Hierarchical Classification:

Top-down Approach: Starts with high-level domains and iteratively narrows down to more specific categories using a hierarchical structure.

10 Bottom-up Approach: Starts with specific concepts and iteratively groups them into broader categories based on their similarities and relationships.

Keyword-based Classification:

Keyword Extraction: Extracts relevant keywords from the input concept using techniques such as TF-IDF, PMI, or TextRank

15 Keyword Matching: Matches the extracted keywords with predefined category labels or ontologies to determine the most appropriate domain or area

Semantic Similarity-based Classification:

20 Embedding-based Similarity: Represents the input concept and category labels as dense vectors using word embeddings or sentence embeddings and calculates their semantic similarity

Knowledge Graph-based Similarity: Leverages external knowledge graphs or ontologies to measure the semantic relatedness between the input concept and category labels.

Confidence Estimation:

25 Probability Scores: Assigns probability scores to each category based on the classification model's output, indicating the model's confidence in the categorization

Distance-based Measures: Calculates distance-based measures, such as cosine distance or Euclidean distance, between the input concept and category representations to quantify the confidence in the categorization

Key Characteristic Identification:

SQ prompts the model to highlight the key aspects or characteristics of the input concept most relevant to mapping it to the target taxonomy.

- 5 SQ employs the following techniques for key characteristic identification:

Named Entity Recognition (NER):

Rule-based NER: Uses predefined rules and patterns to extract named entities, such as persons, organizations, or locations, from the input concept

- 10 Machine Learning-based NER: Applies machine learning models, such as Conditional Random Fields (CRF) or Recurrent Neural Networks (RNN), to identify and classify named entities.

Part-of-Speech (POS) Tagging:

Proofread the provided text:

- 15 Rule-based POS Tagging: Uses predefined rules and heuristics to assign part-of-speech tags, such as nouns, verbs, or adjectives, to each word in the input concept.

Statistical POS Tagging: Employs statistical models, such as Hidden Markov Models (HMM) or Maximum Entropy Markov Models (MEMM), to predict the most likely POS tag for each word.

Dependency Parsing:

- 20 Rule-based Dependency Parsing: Utilizes predefined rules and grammar to analyse the syntactic structure of the input concept and identify the relationships between words.

Neural Dependency Parsing: Uses deep learning models, such as Transition-Based Parsers or Graph-Based Parsers, to predict the dependency structure of the input concept.

Confidence Estimation:

- 25 Probability Scores: Assigns probability scores to each identified characteristic based on the confidence of the NER, POS tagging, or dependency parsing models.

Frequency-based Measures: Calculates frequency-based measures, such as term frequency or inverse document frequency, to assess the importance and relevance of each characteristic.

Target Concept Alignment:

SQ asks the model to determine which concept in the target taxonomy aligns most closely with the input concept based on the identified key characteristics and the model's understanding of the target taxonomy.

- 5 SQ employs the following techniques for target concept alignment:

Semantic Similarity:

Cosine Similarity: Calculates the cosine similarity between the vector representations of the input concept and each target concept to measure their semantic relatedness.

- 10 Euclidean Distance: Computes the Euclidean distance between the vector representations of the input concept and each target concept to assess their semantic proximity.

Ontology Alignment:

Lexical Matching: Matches the lexical terms of the input concept with the labels or synonyms of the target concepts using string similarity measures.

- 15 Structural Matching: Aligns the hierarchical or relational structure of the input concept with the target taxonomy using graph matching or tree alignment algorithms.

Concept Embedding:

Word Embedding-based Alignment: Represents the input concept and target concepts as dense vectors using word embeddings and measures their similarity or distance.

- 20 Sentence Embedding-based Alignment: Encodes the input concept and target concepts as sentence embeddings using models like BERT or RoBERTa and compares their semantic representations.

Confidence Estimation:

Alignment Scores: Assign alignment scores to each target concept based on the semantic similarity, ontology alignment, or concept embedding measures.

- 25 Probability Distribution: Applies a softmax function to the alignment scores to obtain a probability distribution over the target concepts, indicating the model's confidence in the alignment.

Final Prediction and Overall Confidence: SQ asks the model to provide its final prediction for mapping the input concept to the appropriate target concept based on cumulative evidence,

insights gained from previous steps, and an overall confidence score.

SQ employs the following techniques for final prediction and overall confidence:

Evidence Aggregation:

- 5 Weighted Sum: Calculates a weighted sum of the confidence scores or probabilities from the concept categorization, key characteristic identification, and target concept alignment steps.

Bayesian Inference: Applies Bayesian inference techniques to update the belief in each target concept based on accumulated evidence and prior knowledge.

Confidence Calibration:

- 10 Platt Scaling: Calibrates the confidence scores using Platt scaling, which learns a logistic regression model to map the scores to well-calibrated probabilities.

Isotonic Regression: Employs isotonic regression to calibrate the confidence scores by learning a non-parametric monotonic function that maps the scores to probabilities.

Uncertainty Quantification:

- 15 Monte Carlo Dropout: Performs multiple forward passes through the model with dropout enabled to estimate the uncertainty in the final prediction and confidence score.

Ensemble Methods: Combines the predictions and confidence scores from multiple models or multiple runs of the same model using ensemble techniques like bagging or boosting.

Final Prediction Selection:

- 20 Maximum Confidence: Selects the target concept with the highest confidence score or calibrated probability as the final prediction.

Threshold-based Decision: Compares the confidence scores or calibrated probabilities against a predefined threshold to determine the final prediction or defer the decision if the confidence is low.

- 25 Structured Reasoning Process:

SQ follows a structured approach to guide the model's reasoning process in a controlled manner, breaking down the mapping task into smaller, focused questions that enable a step-by-step analysis of the input concept and its alignment with the target taxonomy.

SQ employs the following techniques for structured reasoning:

Question Decomposition:

Hierarchical Questioning: Decomposes the main question into a hierarchy of sub-questions, starting from broad aspects and gradually narrowing down to specific details.

- 5 Dependency-based Questioning: Analyzes the dependencies and relationships between different aspects of the input concept and target taxonomy to generate a sequence of interconnected questions.

Iterative Refinement:

- 10 Progressive Elaboration: Starts with an initial high-level question and iteratively refines or elaborates on the previous answers to gain a deeper understanding of the input concept and its mapping.

Adaptive Questioning: Dynamically adjusts the questioning strategy based on the model's responses, focusing on the most relevant or uncertain aspects to efficiently guide the reasoning process.

Reasoning Chain Tracking:

- 15 Question-Answer Pairing: Keeps track of the question-answer pairs generated during the iterative questioning process to maintain a coherent reasoning chain.

Dependency Graph: Constructs a dependency graph that captures the relationships and dependencies between the questions, answers, and conclusions drawn throughout the reasoning process.

- 20 Confidence Propagation:

Belief Propagation: Propagates the confidence scores or probabilities along the reasoning chain using belief propagation algorithms, such as sum-product or max-product algorithms.

- 25 Uncertainty Propagation: Estimates the uncertainty in the final prediction by propagating the uncertainties associated with each question-answer pair using techniques like Gaussian error propagation or Monte Carlo methods.

Context-Aware Confidence Estimation:

The final confidence estimate provided by SQ considers the cumulative insights gained from the focused questions and responses, reflecting the model's certainty in the final predicted mapping.

SQ generates the context-aware confidence estimation using the following techniques:

Contextual Embedding:

Question-Answer Embedding: Represents each question-answer pair as a dense vector using contextualized word embeddings or sentence embeddings to capture their semantic information.

Reasoning Chain Embedding: Encodes the entire reasoning chain as a single vector by aggregating the question-answer embeddings using techniques like averaging, weighted averaging, or attention mechanisms.

Confidence Aggregation:

Weighted Averaging: Calculates a weighted average of the confidence scores or probabilities associated with each question-answer pair, assigning higher weights to more relevant or informative pairs.

Probabilistic Graphical Models: Employs probabilistic graphical models, such as Bayesian networks or Markov random fields, to aggregate the confidences along the reasoning chain while considering the dependencies and uncertainties.

Contextual Calibration:

Domain Adaptation: Adapts the confidence estimation model to the specific domain or context of the input concept and target taxonomy using domain adaptation techniques like feature augmentation or instance weighting.

Transfer Learning: Leverages pre-trained confidence estimation models from related domains or tasks and fine-tunes them on the specific context using transfer learning techniques like model fine-tuning or meta-learning.

Interpretability and Explainability:

Confidence Attribution: Attributes the final confidence score to the individual question-answer pairs or reasoning steps using techniques like Shapley values or layer-wise relevance propagation (LRP).§ Breadth-First Search (BFS): Performs a breadth-first search on the target taxonomy graph, exploring all concepts at each level before moving to the next level

Concept Matching:

String Matching: Matches the input concept with the concepts in the target taxonomy based on textual similarity measures, such as string-matching algorithms (e.g., Levenshtein distance, Jaccard similarity)

5 Semantic Similarity: Measures the semantic similarity between the input concept and the concepts in the target taxonomy using techniques such as word embeddings, knowledge graph embeddings, or ontology-based methods

Hierarchical Similarity: Considers the hierarchical structure of the target taxonomy to measure the similarity between the input concept and the concepts in the taxonomy based on their shared ancestors or descendants

10 Explanation Generation:

WAP generates an explanation of the reasoning process and the factors contributing to the final confidence estimate using natural language generation techniques.

WAP employs the following techniques for explanation generation:

15 Template-based Generation: Uses predefined templates to generate explanations based on the input concept, the identified keywords, and the matched concept in the target taxonomy

Seq2Seq Models: Trains sequence-to-sequence models on pairs of input concepts and their corresponding explanations to generate explanations for new input concepts. The models can be based on recurrent neural networks (RNNs), transformers, or other sequence-to-sequence architectures.

20 Confidence-based prompting examples:

The COGNIGEN-AX system implements several confidence-based prompting techniques to enhance the AI reasoners' metacognitive abilities and improve the reliability and interpretability of their outputs and confidence estimates. Here are some details on how these techniques are implemented:

25 Chain-of-Thought Prompting (CoTP):

CoTP explicitly prompts the language model to provide a step-by-step reasoning process, explaining its thoughts before giving the final answer and confidence score.

It employs sequence-to-sequence models to generate the chain of thought as a coherent sequence of reasoning steps.

Attention mechanisms like self-attention are used to model the relationships between the reasoning steps.

Techniques like natural language inference and commonsense reasoning are applied to verify the logical consistency of the generated reasoning chain.

- 5 Human evaluation metrics like BLEU and BERTScore are used to assess the quality and coherence of the explanations.

By externalizing the reasoning process, CoTP grounds the confidence estimation in an interpretable chain of logic, enhancing self-awareness and transparency.

Self-Evaluating Prompts (SEP):

- 10 SEP induces a two-stage process where the model first generates an initial prediction, then self-evaluates the correctness of that prediction on a scale (e.g. 0-100) along with an explanation.

It employs uncertainty estimation techniques like Monte Carlo dropout to quantify the model's uncertainty in the self-evaluation score.

- 15 The self-evaluated scores are calibrated using methods like Platt scaling to obtain well-calibrated confidence estimates.

Techniques like SHAP (Shapley Additive explanations) are used to identify the input features that most influenced the self-evaluation score for interpretability.

- 20 SEP enhances calibration by encouraging the model to assess its predictions critically, recognize uncertainties, and provide transparency into its self-assessed confidence levels.

Socratic Questioning (SQ):

SQ systematically guides the model's reasoning through an iterative sequence of focused questions about the input concept and potential mappings to the target domain.

- 25 It decomposes the overall task into sub-questions using hierarchical questioning and dependency analysis techniques.

Based on previous responses, SQ iteratively refines the model's understanding through progressive elaboration and adaptive questioning.

Bayesian inference and belief propagation estimate confidence by aggregating the model's confidence across the question/answer sequence.

SQ fosters step-wise reasoning analysis, enabling the model to identify areas of uncertainty and adjust its confidence in a controlled manner.

By prompting more informative outputs like reasoning chains, self-evaluations, and focused question responses, these techniques enhance the AI reasoners' metacognitive abilities in several ways:

They encourage structured self-reflection on the reasoning process and consideration of potential flaws or uncertainties.

They provide transparent insights into the model's decision factors and confidence assessment rationale.

They facilitate self-criticism by surfacing potential errors or inconsistencies through methods like consistency checking.

They enable self-assessment by quantifying uncertainties and calibrating confidence estimates against the reasoner's judgments.

Ultimately, these introspective prompting techniques lay the groundwork for more self-aware, interpretable, and trustworthy AI systems capable of reliably estimating their confidence levels.

Example:

SARB:ei93:RTG11,% Premium method ranking,ei93,RTG11, Hierarchy for ranking of methods used in provisions assumptions (enumeration),Ratings,,"Method no.1 in ranking,Method no.2 in ranking,Method no.3 in ranking,Method no.4 in ranking,Method no.5 in ranking,Method no.6 in ranking,Method no.7 in ranking,Method no.8 in ranking,Method no.9 in ranking"

Chain-of-Thought Prompting (CoTP):

The system would prompt the language model to walk through its step-by-step reasoning process for mapping the "% Premium method ranking" concept to the metadata attributes.

For example: "The input concept is '% Premium method ranking'. This refers to ranking the percentage premium method among various provisioning methods.

To map this to the metadata attributes:

The Concept would be 'Premium method ranking'

For the description, we need to explain the percentage premium method..."

- This externalized reasoning chain grounds the confidence estimation in interpretable logic and transparency.

Self-Evaluating Prompts (SEP):

5 - The system would first generate an initial mapping like:

"% Premium method ranking" -> Concept: "Premium method ranking", Description: "Ranking of the percentage premium method that calculates provisions as a percentage of earned premiums..."

10 It would then prompt the model to self-evaluate the correctness of this initial mapping on a scale of 0 to 100, along with an explanation.

For example, a self-evaluation of 85 with the explanation, "The description accurately captures the percentage premium method but could provide more context on how the ranking is used."

15 Monte Carlo dropout and techniques like SHAP would quantify uncertainty and interpret which aspects of the input influenced the self-evaluation score.

This enhances calibration by having the model critically assess its output.

Socratic Questioning (SQ):

- The system would guide the mapping process through an iterative sequence of focused questions on the input concept and target metadata schema.

20 For example:

Q1: What type of concept is "% Premium method ranking"?

A1: It relates to an insurance provisioning methodology.

Q2: What specific provisioning method is being referred to?

A2: The percentage premium method for calculating reserves.

25 Q3: What is the significance of the "ranking" aspect?

A3: It indicates this method's relative importance or position compared to others.

...

Bayesian inference techniques would estimate confidence by aggregating the model's confidence across this Q&A sequence.

This step-wise reasoning allows the model to identify uncertain areas and adjust confidence accordingly.

5 Agent: Ensemble of confidence estimation models (ECEM)

Ensemble of Confidence Estimation Models (ECEM) Agent Methods

The ECEM agent leverages an ensemble of confidence estimation models to improve the reliability and accuracy of the confidence estimates for language processing tasks.

10 The ensemble approach involves training multiple confidence estimation models using different techniques and architectures. Each model is responsible for estimating the confidence of a specific aspect or subtask, such as part-of-speech tagging, named entity recognition, or sentiment analysis. The ensemble combines the individual confidence estimates from these models to produce an overall confidence estimate for the task.

ECEM employs the following techniques for confidence estimation:

15 Machine Learning Models:

Classification Models: Trains classification models, such as support vector machines (SVM), random forests, or deep learning models, to estimate the confidence based on input features representing the context, syntactic structure, or semantic information

20 Regression Models: Trains regression models, such as linear regression, decision trees, or neural networks, to predict a continuous confidence score based on input features

Rule-based Approaches:

Expert Rules: Defines expert rules or heuristics based on domain knowledge or linguistic patterns to estimate the confidence

Ensemble Learning:

25 Voting Ensembles: Applies ensemble techniques, such as majority voting or weighted voting, to combine the confidence estimates from multiple models

Stacking Ensembles: Trains a meta-model to learn how to combine the confidence estimates from multiple models based on their performance on a validation set.

Explanation Generation:

ECEM generates human-readable explanations of the reasoning process and the factors contributing to the final confidence estimate using natural language generation techniques.

ECEM employs the same techniques for explanation generation as the WAP agent, including template-based generation and seq2seq models. § Breadth-First Search (BFS):

- 5 Performs a breadth-first search on the target taxonomy graph, starting from the root concept and exploring all concepts at each level before moving to the next level.

Concept Similarity:

- 10 Semantic Similarity: Calculates the semantic similarity between the input concept and each target concept using techniques such as cosine similarity, Jaccard similarity, or semantic embeddings (e.g., Word2Vec, GloVe).

Hierarchical Distance: Measures the hierarchical distance between the input concept and each target concept in the taxonomy using techniques such as the shortest path length or the Wu-Palmer similarity.

Concept Embedding:

- 15 Word Embeddings: Represents each concept as a dense vector using pre-trained word embeddings such as Word2Vec, GloVe, or FastText.

Contextualized Embeddings: Generates contextualized embeddings for each concept using pre-trained language models such as BERT, RoBERTa, or ELMo, capturing the semantic information in the context of the taxonomy.

- 20 Confidence Estimates and Predicted Mappings:

- WAP obtains confidence estimates and predicted mappings from various techniques, such as Verbalized Confidence, Token Probabilities, Prompt Entropy, Semantic Output Clustering, Self-Consistency, Chain-of-Thought, Self-Evaluating, and Socratic Questioning. Each technique provides its own prediction and confidence score based on its specific approach to confidence estimation.

WAP employs the following techniques for obtaining confidence estimates and predicted mappings:

Verbalized Confidence:

- 30 Explicit Confidence Statements: Extracts explicit confidence statements from the language model's output using pattern matching or regular expressions.

Confidence Scoring: Assigns numerical confidence scores to the verbalized confidence statements using a predefined scale or mapping function.

Token Probabilities:

5 Probability Extraction: Extracts the token probabilities from the language model's output distribution for each predicted mapping.

Confidence Aggregation: Aggregates the token probabilities into a single confidence score using techniques such as averaging, weighted averaging, or product of probabilities.

Prompt Entropy:

10 Entropy Calculation: Calculates the entropy of the language model's output distribution when prompted with different target concepts.

Confidence Mapping: Maps the entropy values to confidence scores using a predefined function or learned mapping.

Semantic Output Clustering:

15 Embedding-based Clustering: Applies clustering algorithms (e.g., k-means, hierarchical clustering) to the semantic embeddings of the predicted mappings.

Cluster Confidence Assignment: Assign confidence scores to each cluster based on the coherence or compactness of the cluster.

Self-Consistency:

20 Consistency Evaluation: Evaluates the consistency of predicted mappings across multiple runs or variations of the input concept. After calculating the weighted average of the predicted mappings, the resulting average can be further normalized to ensure it falls within a desired range (e.g., [0, 1]).

Normalization techniques, such as min-max normalization or z-score normalization, can be applied to adjust the weighted average accordingly.

25 The final prediction is typically determined based on the normalized weighted average, with mappings having higher average values being considered as more likely predictions.

The confidence score of the final prediction can also be derived from the normalized weighted average, providing an indication of the model's confidence in its prediction.

It is important to note that the specific normalization and prediction methods used may vary depending on the application and requirements of the language model. WAP normalizes the weighted average result to obtain the final predicted mapping, ensuring that it represents a valid mapping within the target taxonomy.

5 WAP employs the following techniques for normalization and final prediction:

Softmax Normalization:

Exponential Transformation: Applies the exponential function to the weighted average scores, mapping the scores to positive values.

10 Softmax Function: Normalizes the exponentiated scores using the softmax function, converting the scores into a probability distribution over the target concepts.

Argmax Selection:

Maximum Probability: Selects the target concept with the highest probability in the normalized distribution as the final predicted mapping.

15 Confidence Thresholding: Applies a confidence threshold to the normalized probabilities, considering only mappings above a certain confidence level as valid predictions.

Mathematical Formulation:

Let $\bar{m} = (\bar{m}_1, \bar{m}_2, \dots, \bar{m}_k)$ be the weighted average scores for each target concept.

20 The softmax normalization of the weighted average scores, denoted as p_i , is calculated as:

$$p_i = \frac{e^{\bar{m}_i}}{\sum_{j=1}^k e^{\bar{m}_j}}$$

The final predicted mapping, denoted as m^* , is selected as:

$$m^* = \operatorname{argmax}_{i \in \{1, 2, \dots, k\}} p_i$$

25

Prioritization of Accurate Predictions:

WAP prioritizes accurate model predictions while suppressing noisy or low-confidence outputs by using the confidence scores as weights in the averaging process. Models with higher confidence in a particular input have a stronger influence on the ensemble prediction, while less confident models have a diminished impact.

30

WAP achieves the prioritization of accurate predictions through the following techniques:

Confidence-based Weighting:

Higher weights are assigned to predictions with higher confidence scores, allowing them to dominate the weighted average calculation.

5 Lower weights are assigned to predictions with lower confidence scores, reducing their influence on the final prediction.

Confidence Thresholding:

A minimum confidence threshold is applied to filter out predictions with low confidence scores, eliminating noisy or unreliable outputs from the ensemble.

10 Only predictions above the confidence threshold are considered in the weighted average calculation, ensuring that the final prediction is based on reliable and confident outputs.

Adaptive Weighting Schemes:

The weighting scheme can be adapted based on the characteristics of the input concept or the performance of individual models.

15 For example, if a particular model consistently provides accurate predictions for a specific domain or concept type, its confidence scores can be given higher weights in the averaging process.

Ensemble Pruning:

Models with consistently low confidence scores or poor performance can be pruned from the ensemble, reducing the impact of noisy or unreliable models. § Pruning techniques, such as
20 confidence-based pruning or accuracy-based pruning, can be applied to remove underperforming models from the ensemble.

Robustness and Reliability:

WAP enhances the robustness and reliability of the final prediction by leveraging the collective knowledge and confidence estimates from multiple techniques. By combining
25 predictions from diverse approaches, WAP reduces the impact of individual model biases or limitations, leading to more accurate and well-rounded predictions.

WAP achieves robustness and reliability through the following techniques:

Diversity in Ensemble Techniques:

Incorporating a diverse set of confidence estimation techniques, such as Verbalized
30 Confidence, Token Probabilities, Prompt Entropy, Semantic Output Clustering, Self-Consistency, Chain-of-Thought, Self-Evaluating, and Socratic Questioning.

Each technique brings a unique perspective and approach to confidence estimation, capturing different aspects of the language model's predictions.

Complementary Strengths:

5 Leveraging the complementary strengths of different techniques, where some methods may excel in certain domains or concept types while others perform better in different contexts.

The ensemble approach allows the strengths of individual techniques to compensate for the weaknesses of others, resulting in a more balanced and reliable prediction.

Confidence Score Calibration:

10 Applying calibration techniques, such as Platt scaling or isotonic regression, to ensure that the confidence scores from different techniques are well-calibrated and comparable.

Calibration helps align the confidence scores with the actual accuracy or reliability of the predictions, improving the overall quality of the weighted average.

Ensemble Size and Composition:

15 Determining the optimal size and composition of the ensemble based on factors such as the diversity of techniques, the computational complexity, and the desired level of robustness.

Balancing the trade-off between the benefits of including more techniques and the potential overhead or diminishing returns of a larger ensemble.

Context-Aware Confidence Estimation:

20 The final output of WAP includes the ensemble predicted mapping and its weighted average confidence score. This output provides a context-aware confidence estimation that considers the predictions and scores from multiple techniques, considering their collective confidence assessment in the specific input concept and target taxonomy.

WAP generates the context-aware confidence estimation using the following techniques:

Contextual Embedding:

25 Representing the input concept and target concepts as dense vectors using contextual embedding techniques, such as BERT or ELMo.

Capturing the semantic information and relationships between concepts in the specific context of the input and target domains.

Domain-Specific Confidence Calibration:

Calibrating the confidence scores of individual techniques using domain-specific data or knowledge.

Adjusting the confidence scores based on the characteristics and requirements of the domain or taxonomy.

5 Contextual Weighting:

Assigning weights to individual techniques based on their performance or relevance in the specific context of the input concept and target taxonomy.

Giving higher weights to techniques that have demonstrated strong performance or domain expertise in similar contexts.

10 Confidence Score Interpretation:

Providing an interpretation or explanation of the final weighted average confidence score in the context of the input concept and target taxonomy.

Highlighting the key factors or techniques that contributed to the confidence estimation and their relative importance in the specific context.

15 Agent: Threshold voting:

The TV-CMV agent generates a final predicted mapping by treating each model prediction as a "vote" weighted by its confidence score. This ensembling technique employs a complex model selection process that selects the ensemble prediction with the maximum weighted vote total. TV-CMV leverages the collective decision-making of multiple models while prioritizing high-confidence predictions.

Concept Analysis:

TV-CMV begins by prompting the model to analyze the input concept and its definition, identifying the key characteristics and criteria that define the concept.

TV-CMV employs the following techniques for concept analysis:

25 Semantic Parsing:

Dependency Parsing: Applies dependency parsing to extract the syntactic structure of the input concept definition, identifying the relationships between words and phrases.

Semantic Role Labeling: Performs semantic role labeling to identify the semantic arguments and their roles in the input concept definition, such as the agent, patient, or instrument.

Named Entity Recognition:

Entity Extraction: Extracts named entities from the input concept definition using techniques such as rule-based matching, machine learning models (e.g., Conditional Random Fields, Recurrent Neural Networks), or pre-trained named entity recognizers.

- 5 Entity Linking: Links the extracted entities to their corresponding entries in a knowledge base or ontology using techniques such as string matching, semantic similarity, or graph-based methods.

Keyword Extraction:

- 10 Statistical Approaches: Applies statistical methods, such as TF-IDF (Term Frequency-Inverse Document Frequency) or PMI (Pointwise Mutual Information), to identify the most informative and relevant keywords in the input concept definition.

- 15 Graph-based Approaches: Constructs a graph representation of the input concept definition, where nodes represent words and edges represent their co-occurrence or semantic relationships, and applies centrality measures (e.g., PageRank, TextRank) to identify the most important keywords.

Target Taxonomy Review:

TV-CMV prompts the model to review the target taxonomy and its concepts to thoroughly understand the options for mapping the input concept.

TV-CMV employs the following techniques for target taxonomy review:

- 20 Taxonomy Traversal:

Depth-First Search (DFS): Performs a depth-first search on the target taxonomy graph, starting from the root concept and exploring each branch to its leaf concepts.

- 25 Breadth-First Search (BFS): Performs a breadth-first search on the target taxonomy graph, starting from the root concept and exploring all concepts at each level before moving to the next level.

Concept Similarity:

Semantic Similarity: This calculates the semantic similarity between the input concept and each target concept using techniques such as cosine similarity, Jaccard similarity, or semantic embeddings (e.g., Word2Vec, GloVe).

Hierarchical Distance: This measures the hierarchical distance between the input concept and each target concept in the taxonomy using techniques such as the shortest path length or the Wu-Palmer similarity.

Concept Embedding:

- 5 Word Embeddings: This represents each concept as a dense vector using pre-trained word embeddings such as Word2Vec, GloVe, or FastText.

Contextualized Embeddings: This generates contextualized embeddings for each concept using pre-trained language models such as BERT, RoBERTa, or ELMo, capturing the semantic information in the context of the taxonomy.

- 10 Confidence Estimates and Predicted Mappings:

TV-CMV obtains confidence estimates and predicted mappings from various techniques, such as Verbalized Confidence, Token Probabilities, Prompt Entropy, Semantic Output Clustering, Self-Consistency, Chain-of-Thought, Self-Evaluating, and Socratic Questioning.

- 15 Each technique provides its own prediction and confidence score based on its specific approach to confidence estimation.

TV-CMV employs the following techniques for obtaining confidence estimates and predicted mappings:

Verbalized Confidence:

- 20 Explicit Confidence Statements: This extracts explicit confidence statements from the language model's output using pattern matching or regular expressions.

Confidence Scoring: This assigns numerical confidence scores to the verbalized confidence statements using a predefined scale or mapping function.

Token Probabilities:

- 25 Probability Extraction: This extracts the token probabilities from the language model's output distribution for each predicted mapping.

Confidence Aggregation: This aggregates the token probabilities into a single confidence score using techniques such as averaging, weighted averaging, or product of probabilities.

Prompt Entropy:

Entropy Calculation: This calculates the entropy of the language model's output distribution when prompted with different target concepts.

Confidence Mapping: This maps the entropy values to confidence scores using a predefined function or learned mapping.

5 Semantic Output Clustering:

Embedding-based Clustering: This applies clustering algorithms (e.g., k-means, hierarchical clustering) to the semantic embeddings of the predicted mappings.

Cluster Confidence Assignment: This assigns confidence scores to each cluster based on the coherence or compactness of the cluster.

10 Self-Consistency:

Consistency Evaluation: This evaluates the consistency of predicted mappings across multiple runs or variations of the input concept.

Consistency-based Confidence: This calculates confidence scores based on the degree of consistency or agreement among the predicted mappings.

15 Chain-of-Thought:

Step-by-Step Reasoning: Generates step-by-step reasoning chains for each predicted mapping, explaining the rationale behind the prediction.

Reasoning Quality Assessment: Assesses the quality and coherence of the reasoning chains and assigns confidence scores based on predefined criteria or learned models.

20 Self-Evaluating:

Self-Evaluation Prompts: Prompts the language model to evaluate the correctness of its own predicted mappings and provide confidence scores.

Calibration and Adjustment: Calibrates and adjusts the self-evaluated confidence scores using techniques such as Platt scaling or isotonic regression.

25 Socratic Questioning:

Iterative Questioning: Engages in an iterative process of asking focused questions to guide the language model toward a final prediction and confidence estimate.

Confidence Aggregation: Aggregates the confidence scores obtained from each questioning step into a final confidence estimate using techniques such as weighted averaging or Bayesian updating.

Weighted Voting:

- 5 TV-CMV treats each predicted mapping as a "vote" weighted by its confidence score. The weight assigned to each vote is determined by the confidence score associated with the corresponding prediction.

TV-CMV employs the following techniques for weighted voting:

Vote Representation:

- 10 Binary Vote Vector: Represents each predicted mapping as a binary vote vector, where a value of 1 indicates a vote for the corresponding target concept, and 0 indicates no vote.

Confidence Score Vector: Represents the confidence scores associated with each predicted mapping as a vector of floating-point numbers.

Vote Weighting:

- 15 Confidence-based Weights: Assigns weights to each vote based on the confidence score of the corresponding prediction.

Logarithmic Scaling: Applies logarithmic scaling to the confidence scores before assigning weights to handle imbalanced confidence distributions.

Mathematical Formulation:

- 20 Let $M = \{m_1, m_2, \dots, m_n\}$ be the set of predicted mappings and $C = \{c_1, c_2, \dots, c_n\}$ be the corresponding confidence scores.

The weighted vote for a target concept t is calculated as:

$$v_t = \sum_{i=1}^n w_i \times \mathbb{1}(m_i = t)$$

- 25 where w_i is the weight assigned to the i -th prediction based on its confidence score, and $\mathbb{1}(m_i = t)$ is an indicator function that equals 1 if the i -th prediction maps to target concept t and 0 otherwise.

Weighted Vote Summation:

TV-CMV calculates the sum of the weighted votes for each candidate mapping, determining the total support for each mapping based on the confidence-weighted votes.

TV-CMV employs the following techniques for weighted vote summation:

Vote Accumulation:

- 5 Initialization: Initializes a vote accumulator for each target concept, setting their initial values to zero. Vote Addition: The algorithm iterates over the weighted votes and adds the corresponding weight to the vote accumulator of the target concept associated with each vote.

Mathematical Formulation:

- 10 Let $T = \{t_1, t_2, \dots, t_k\}$ be the set of target concepts and $V = \{v_1, v_2, \dots, v_k\}$ be the corresponding vote accumulators.

The weighted vote sum for a target concept t_j is calculated as:

$$s_j = \sum (w_i * 1(m_i = t_j))$$

- 15 where w_i is the weight assigned to the i -th prediction based on its confidence score, and $1(m_i = t_j)$ is an indicator function that equals 1 if the i -th prediction maps to target concept t_j and 0 otherwise.

The vote accumulation process results in a vector of weighted vote sums $S = \{s_1, s_2, \dots, s_k\}$ for each target concept.

Maximum Weighted Vote Selection:

- 20 The final prediction is selected as the mapping with the maximum weighted vote total, considering the collective decision of the models based on their confidence-weighted votes.

TV-CMV employs the following techniques for maximum weighted vote selection:

Argmax Operation:

- 25 Comparison: Compares the weighted vote sums of all target concepts and identifies the target concept with the highest vote sum.

Tie-breaking: Applies a predefined tie-breaking strategy (e.g., random selection, lexicographic order) if multiple target concepts have the same maximum vote sum.

Mathematical Formulation:

Let $S = \{s_1, s_2, \dots, s_k\}$ be the vector of weighted vote sums for each target concept.

The final predicted mapping m^* is selected as:

$$m^* = \operatorname{argmax}(t \in T) s_t$$

where T is the set of target concepts and s_t is the weighted vote sum for target concept t .

- 5 If multiple target concepts have the same maximum vote sum, a tie-breaking strategy is applied to select a single mapping as the final prediction.

Prioritization of High-Confidence Predictions:

- TV-CMV prioritizes high-confidence predictions in the ensemble estimation by weighting the votes based on the confidence scores. Models with higher confidence in a particular mapping contribute more to the weighted vote total, increasing the likelihood of that mapping being selected as the final prediction.
- 10

TV-CMV achieves the prioritization of high-confidence predictions through the following techniques:

Confidence-based Weighting:

- 15 Higher weights are assigned to votes with higher confidence scores, allowing them to have a stronger influence on the final prediction.

Lower weights are assigned to votes with lower confidence scores, reducing their impact on the ensemble decision.

Confidence Thresholding:

- 20 A minimum confidence threshold is applied to filter out votes with low confidence scores, eliminating unreliable or noisy predictions from the voting process.

Only votes above the confidence threshold are considered in the weighted vote summation, ensuring that the final prediction is based on high-confidence mappings.

Adaptive Weighting Schemes:

- 25 The weighting scheme can be adapted based on the characteristics of the input concept or the performance of individual models.

For example, if a particular model consistently provides high-confidence predictions for a specific domain or concept type, its votes can be given higher weights in the voting process.

Ensemble Pruning:

Models with consistently low confidence scores or poor performance can be pruned from the ensemble, reducing the impact of unreliable or noisy models.

5 Pruning techniques, such as confidence-based or accuracy-based pruning, can be applied to remove underperforming models from the voting process.

Robustness and Reliability:

10 TV-CMV enhances the final prediction's robustness and reliability by leveraging the collective decision-making of multiple models. By considering the weighted votes from diverse techniques, TV-CMV reduces the influence of individual model biases or limitations, leading to more accurate and well-supported predictions.

TV-CMV achieves robustness and reliability through the following techniques:

Diversity in Voting Techniques:

15 Incorporating a diverse set of confidence estimation techniques, such as Verbalized Confidence, Token Probabilities, Prompt Entropy, Semantic Output Clustering, Self-Consistency, Chain-of-Thought, Self-Evaluating, and Socratic Questioning.

Each technique contributes a unique perspective and approach to confidence estimation, capturing different aspects of the language model's predictions.

Majority Voting Principle:

20 Leveraging the principle of majority voting, where the collective decision of multiple models is considered more reliable than individual model predictions.

The weighted voting process allows the consensus of high-confidence predictions to outweigh the influence of noisy or outlier predictions.

Confidence Score Calibration:

25 Applying calibration techniques, such as Platt scaling or isotonic regression, to ensure that the confidence scores from different techniques are well-calibrated and comparable.

Calibration helps align the confidence scores with the actual accuracy or reliability of the predictions, improving the quality of the weighted voting process.

Ensemble Size and Composition:

Determining the optimal size and composition of the ensemble based on factors such as the diversity of techniques, the computational complexity, and the desired level of robustness.

Balancing the trade-off between the benefits of including more techniques and the potential overhead or diminishing returns of a larger ensemble.

5 Context-Aware Confidence Estimation:

The TV-CMV final output comprises of the ensemble predicted mapping and its weighted vote total. This output provides a confidence estimation that considers predictions and scores from multiple techniques. It considers their collective assessment of confidence in the specific input concept and target taxonomy.

10

TV-CMV generates the context-aware confidence estimation using the following techniques:

Contextual Embedding:

Representing the input concept and target concepts as dense vectors using contextual embedding techniques like BERT or ELMo.

15 Capturing the semantic information and relationships between concepts in the context of the input and target domains.

Domain-Specific Confidence Calibration:

Calibrating the confidence scores of individual techniques using domain-specific data or knowledge.

20 Adjusting the confidence scores based on the characteristics and requirements of the particular domain or taxonomy.

Contextual Weighting:

Assigning weights to individual techniques based on their performance or relevance in the context of the input concept and target taxonomy.

25 Giving higher weights to techniques that have demonstrated strong performance or domain expertise in similar contexts.

Confidence Score Interpretation:

Providing an interpretation or explanation of the final weighted vote total in the context of the input concept and target taxonomy.

30 Highlighting the key factors or techniques that contributed to the confidence estimation and their relative importance in the specific context.

Agent: Nash Embedding for Confidence Integration

Agent Methods

The NECI agent is a game-theoretic approach that combines confidence estimates from multiple techniques in a principled and self-adjusting manner. By reframing the integration problem as a decentralized mechanism design challenge using non-cooperative games, NECI allows confidence estimation techniques to specialize in different input patterns dynamically. At the same time, the referee module learns to optimally compose them into a well-calibrated ensemble.

Concept Analysis:

- NECI begins by directing the model to analyze the input concept and its definition, identifying key characteristics and criteria that define the concept.

NECI employs the following techniques for concept analysis:

Semantic Parsing:

- Dependency Parsing: Extracts the syntactic structure of the input concept definition, identifying relationships between words and phrases.

Semantic Role Labeling: Identifies semantic arguments and their roles in the input concept definition, such as the agent, patient, or instrument.

Named Entity Recognition:

- Entity Extraction: Extracts named entities from the input concept definition using techniques like rule-based matching, machine learning models (e.g., Conditional Random Fields, Recurrent Neural Networks), or pre-trained named entity recognizers.

Entity Linking: Links the extracted entities to their corresponding entries in a knowledge base or ontology using techniques like string matching, semantic similarity, or graph-based methods.

- Keyword Extraction:

Statistical Approaches: Identifies the most informative and relevant keywords in the input concept definition using statistical methods like TF-IDF (Term Frequency-Inverse Document Frequency) or PMI (Pointwise Mutual Information).

- Graph-based Approaches: Constructs a graph representation of the input concept definition, with nodes representing words and edges representing their co-occurrence or semantic

relationships. Applies centrality measures (e.g., PageRank, TextRank) to identify the most important keywords.

Target Taxonomy Review:

- NECI directs the model to thoroughly review the target taxonomy and its concepts to
5 understand all the options for mapping the input concept.

NECI employs the following techniques for target taxonomy review:

Taxonomy Traversal:

- Depth-First Search (DFS): Performs a depth-first search on the target taxonomy graph, starting from the root concept and exploring each branch to its leaf concepts.
- 10 Breadth-First Search (BFS): Performs a breadth-first search on the target taxonomy graph, starting from the root concept and exploring all concepts at each level before moving to the next level.

Concept Similarity:

- Semantic Similarity: Calculates the semantic similarity between the input concept and each
15 target concept using techniques such as cosine similarity, Jaccard similarity, or semantic embeddings (e.g., Word2Vec, GloVe).

Hierarchical Distance: Measures the hierarchical distance between the input concept and each target concept in the taxonomy using techniques such as the shortest path length or the Wu-Palmer similarity.

- 20 Concept Embedding:

Word Embeddings: Represents each concept as a dense vector using pre-trained word embeddings such as Word2Vec, GloVe, or FastText.

- Contextualized Embeddings: Generates contextualized embeddings for each concept using pre-trained language models such as BERT, RoBERTa, or ELMo, capturing the semantic
25 information in the context of the taxonomy.

Game Definition:

NECI defines each confidence estimation technique as a self-interested agent in a non-cooperative game, aiming to maximize its own accuracy while contributing to the ensemble's overall performance.

NECI employs the following techniques for game definition:

Agent Definition:

Technique Representation: Represents each confidence estimation technique, such as Verbalized Confidence, Token Probabilities, Prompt Entropy, Semantic Output Clustering, and Self-Consistency, as a separate agent in the game.

Agent Strategies: Defines the strategy space for each agent, which includes the possible mappings and confidence scores they can produce.

Utility Functions:

Accuracy Maximization: Assigns a utility function to each agent that rewards accurate predictions and penalizes incorrect ones, encouraging the agents to maximize their accuracy.

Ensemble Contribution: Incorporates a term in the utility function that rewards agents for contributing to the overall accuracy and calibration of the ensemble, promoting cooperation and specialization.

Nash Equilibrium:

Best Response Dynamics: Applies best response dynamics, where each agent iteratively updates its strategy to maximize its utility given the current strategies of the other agents.

Convergence: Iterates the best response dynamics until convergence to a Nash equilibrium, where no agent can unilaterally improve its utility by changing its strategy.

Referee Introduction:

NECI introduces a referee agent that learns an optimal composition of the confidence estimation techniques based on their performance and specialization.

NECI employs the following techniques for referee introduction:

Referee Objective:

Ensemble Accuracy: Defines the objective of the referee agent as maximizing the overall accuracy and calibration of the ensemble by composing the individual techniques optimally.

Specialization Encouragement: Incorporates a term in the referee's objective that encourages specialization among the methods, promoting their focus on different input patterns and mapping categories.

Composition Learning:

Proofreading - improving clarity and flow:

Reinforcement Learning: Reinforcement learning techniques, such as Q-learning or policy gradients, are applied to learn the optimal composition strategy for the referee agent.

- 5 Reward Function: A reward function is designed for the referee agent, considering the accuracy and calibration of the ensemble, as well as the specialization and cooperation among the techniques.

Adaptive Composition:

- 10 Dynamic Adjustment: The referee agent can dynamically adjust the composition of techniques based on the characteristics of the input concept and the performance of the individual techniques.

- Exploration-Exploitation Trade-off: The exploration of new composition strategies is balanced with the exploitation of proven strategies to adapt to changing input patterns and
- 15 maintain optimal performance.

Payoff Matrix Construction:

NECI constructs payoff matrices to represent the relative accuracies or rewards of each technique for different mappings. Factors such as input concept complexity, target taxonomy size, and output quality are considered.

- 20 NECI employs the following techniques for payoff matrix construction:

Mapping Categorization:

Complexity Scoring: Complexity scores are assigned to input concepts based on factors such as the number of attributes, relationships, or semantic depth, capturing the difficulty of mapping the concept.

- 25 Taxonomy Size Scoring: The size and granularity of the target taxonomy are measured using metrics such as the number of concepts, depth, or branching factor, indicating the available mapping options.

- Output Quality Assessment: The quality of the generated mappings is evaluated using metrics such as semantic similarity, coherence, or domain relevance, reflecting the
- 30 effectiveness of the techniques.

Payoff Calculation:

Accuracy Measurement: The accuracy of each technique is calculated for different combinations of complexity scores, taxonomy size scores, and output quality scores using a representative validation set.

- 5 Reward Assignment: Rewards are assigned to each technique based on their relative performance compared to other techniques, capturing their strengths and weaknesses for different mapping categories.

Matrix Formulation:

- 10 Dimension Definition: The dimensions of the payoff matrices are defined based on the number of techniques and the granularity of the mapping categories (e.g., complexity levels, taxonomy sizes, quality levels).

Entry Population: The entries of the payoff matrices are populated with the calculated rewards for each technique and mapping category combination, creating a comprehensive representation of their relative performance.

- 15 Nash Equilibrium Computation:

NECI computes the Nash equilibrium of the game using the Lemke-Howson algorithm, which guarantees the existence of at least one equilibrium in a non-cooperative game.

NECI employs the following techniques for Nash equilibrium computation:

Lemke-Howson Algorithm: Proofread this for clarity and flow:

- 20 Tableau Construction: Constructs a tableau representation of the game using the payoff matrices. This captures the strategies and utilities of each agent.

Pivoting Steps: Performs a series of pivoting steps in the tableau, following the Lemke-Howson algorithm. This is done to find a complementary solution.

- 25 Equilibrium Extraction: Extracts the Nash equilibrium strategies for each agent from the final tableau. These strategies represent their optimal strategies given the strategy of the other agents.

Equilibrium Interpretation:

Strategy Analysis: Analyzes the equilibrium strategies to understand the specialization and cooperation patterns among the techniques.

Composition Implications: Interprets the equilibrium strategies in terms of the optimal composition of techniques for different mapping categories. This provides insights into their relative strengths and weaknesses.

Equilibrium Adaptation:

- 5 Dynamic Updating: Updates the equilibrium strategies dynamically as new input concepts and mapping categories are encountered. This allows the techniques to adapt their specialization and cooperation patterns.

- 10 Continuous Learning: Incorporates continuous learning mechanisms for the referee agent to refine its composition strategy. This is based on the evolving equilibrium and the performance of the ensemble.

Mapping Assignment:

Based on the equilibrium strategy profile, NECI assigns the final mapping by combining the confidence estimates from each technique using a weighted aggregation scheme.

- 15 NECI employs the following techniques for mapping assignments:

Confidence Score Aggregation:

Weighted Averaging: Calculates a weighted average of the confidence scores from each technique, where the equilibrium strategy profile determines the weights.

- 20 Geometric Mean: Computes the geometric mean of the confidence scores. This effectively balances the contributions of each method and is less sensitive to outliers compared to arithmetic mean.

Mapping Selection:

Argmax Selection: Selects the target concept with the highest aggregated confidence score as the final mapping.

- 25 Threshold-based Filtering: Applies a confidence threshold to filter out mappings with low confidence scores. This ensures the selected mapping meets a minimum confidence level.

Mapping Interpretation:

Confidence Score Analysis: Analyses the individual confidence scores from each technique to understand their relative contributions to the final mapping decision.

- 30 Mapping Justification: Justifies the selected mapping based on the equilibrium strategy profile. This highlights the key techniques and their specialization patterns that influenced the decision.

Payoff Matrix Construction: The payoff matrix is a crucial component of the game-theoretic formulation. It encodes the incentives for each technique to specialize in different mapping categories based on their empirical accuracies. NECI constructs the payoff matrix by categorizing the domain mappings based on relevant features such as the types of data element concepts being mapped, the complexity of the mapping description, or other characteristics that may influence the performance of confidence estimation techniques. The accuracy or reward obtained by each method for each mapping category is empirically evaluated using a representative validation set and recorded in the payoff matrix.

Nash Equilibrium Computation:

- 10 With the payoff matrix in hand, NECI computes the Nash equilibrium of the game, representing the optimal strategy profile for composing the techniques. The Nash equilibrium is formulated as a linear complementarity problem (LCP) and solved using techniques such as the Lemke-Howson algorithm or linear programming. The equilibrium solution is a probability distribution over the confidence estimation techniques, indicating the proportion of mappings assigned to each method based on their specialization.
- 15

Referee Module:

- NECI integrates the game-theoretic approach into the overall confidence estimation pipeline through a referee module. This module acts as a central authority to coordinate the composition of techniques based on the computed Nash equilibrium. The referee module inputs the equilibrium strategy and assigns incoming mappings to the appropriate confidence estimation techniques according to their assigned proportions. It collects the individual confidence scores from each method and combines them using a weighted aggregation scheme, where the equilibrium proportions determine the weights.
- 20

Dynamic Specialization and Ensemble Calibration:

- 25 Leveraging the game-theoretic approach, NECI enables confidence estimation techniques to specialize in different input patterns dynamically. The methods are incentivized to focus on mapping categories where they exhibit higher accuracy, leading to improved overall performance. At the same time, the referee module learns to compose the techniques optimally into a well-calibrated ensemble, ensuring that the final confidence estimates are reliable and well-aligned with the actual mapping accuracies.
- 30

Interoperability and Standardization:

NECI uses structured prompts and JSON format to ensure interoperability and standardization. The input concept, target taxonomy, and confidence estimates from multiple techniques are provided in a systematic way, making it easier to integrate the Nash Embedding approach with the broader COGNATE system, as well as other methods and systems. The final output, which includes the predicted mapping and the summary confidence score, is also presented in a standardized JSON format, allowing for seamless communication and integration with other components.

ISOCLES

ISOCLES (Iterative Self-Optimization through Continual Learning and Evaluation Synthesis) is a comprehensive workflow that integrates the REIMAS and IPASE methodologies into a unified system and methods for continuous AI optimization. ISOCLES enables AI systems to engage in structured self-reflection, external validation, iterative refinement, and targeted skill development, empowering them to autonomously optimize their performance and learn from real-world interactions and feedback.

Agents

LITE: Logging and instrumentation agent that creates standardized logs capturing all execution details such as net transitions, token movements, agent inputs/outputs, and errors.

AIKA: Prompt analysis and enhancement agent that optimizes initial prompts by understanding user context, evaluating ethics, preserving core intent, and adding clarifying details.

MOPT: Multi-agent orchestration through an executable Petri Nets agent that controls execution flow, agent interactions, and state transitions, adhering to ISO 15909.

Evaluation & Validation

Process Traceability: Comprehensive logging by LITE enables full traceability of ISOCLES execution, with detailed logs capturing net transitions, data flows, and errors for analysis.

Prompt Refinement Assessment: AIKA's prompt enhancements were evaluated through user studies on relevance, ethical compliance, and context-matching of enhanced prompts.

Formal Methods Verification: Using model-checking techniques, MOPT's Petri Net models were formally verified for liveness, boundedness, and other properties.

ISOCLES is a comprehensive and integrated workflow engine that orchestrates the various modules and agents of COGNIGEN-AX. ISOCLES establishes a structured and iterative approach that guides AI reasoners through cycles of self-reflection, external validation, iterative refinement, and targeted skill development. By defining clear interfaces, data flows, and control mechanisms, ISOCLES enables seamless integration and synchronization of their respective functionalities. The ultimate objective is to create a holistic optimization pipeline that empowers AI reasoners to enhance their performance autonomously, adapt to evolving requirements, and maintain high reliability and effectiveness over time.

Capability:

Modular and hierarchical multi-agent orchestration.

ISOCLES incorporates a modular and hierarchical multi-agent orchestration capability. This involves designing a flexible and extensible architecture that allows for the composition and coordination of multiple AI agents, each representing a specific functionality or system and methods (e.g., COGNIGEN-AX, REIMAS, COGNATE, IPASE). The orchestration mechanism defines clear interfaces and communication protocols between the agents, enabling them to exchange data and control signals and feedback in a structured and standardized manner. ISOCLES leverages well-established orchestration patterns and technologies, such as service-oriented architectures, event-driven architectures, or workflow management systems, to ensure the overall system's scalability, reliability, and maintainability. Additionally, the orchestration capability supports hierarchical decomposition and modularization, allowing for the encapsulation of complex functionalities within higher-level agents and the reuse of standard components across different workflows.

Configurable and adaptable optimization pipelines

ISOCLES system and methods enable the customization and fine-tuning of the optimization process based on specific application contexts, user requirements, and performance objectives. This involves providing a set of configurable parameters, templates, and rules that allow users to define the desired behavior and outcomes of the optimization workflow. ISOCLES supports the specification of optimization goals, quality metrics, convergence criteria, and resource constraints, among other factors. Additionally, ISOCLES incorporates mechanisms for dynamically adapting the pipeline configuration based on runtime feedback, performance measurements, or changes in the operating environment. This may involve reinforcement learning, Bayesian optimization, or evolutionary algorithms automatically tuning the pipeline parameters and selecting the most effective optimization strategies. ISOCLES can provide the flexibility and customization necessary to address diverse

optimization scenarios and evolving requirements by enabling configurable and adaptable optimization pipelines.

Monitoring and analytics for optimization lifecycle

ISOCLES offers visibility, transparency, and insights into the optimization lifecycle by instrumenting the various components of the optimization pipeline to collect relevant performance metrics, quality indicators, and runtime statistics. ISOCLES defines standardized monitoring and logging methods that capture the inputs, outputs, intermediate results, and decision points of each optimization step. Additionally, ISOCLES incorporates advanced analytics and visualization techniques to process and interpret the collected data, allowing users to gain actionable insights into the optimization process. This can include techniques like anomaly detection, root cause analysis, and predictive modeling to identify bottlenecks, diagnose issues, and forecast optimization outcomes. By providing comprehensive monitoring and analytics capabilities, ISOCLES empowers users to assess the effectiveness of the optimization workflow, identify areas for improvement, and make data-driven decisions to continuously enhance the performance and reliability of AI reasoners.

FIG. 4 is a flowchart 400 illustrating an example process for managing a multi-agent system within an artificial intelligence system, according to an embodiment herein. This flowchart 400 details the operational steps designed to enhance decision-making capabilities by coordinating multiple agents. Below is an overview of each step involved in the process:

- Initialize Multi-Agent System 402: This step involves setting up the multi-agent system, which may include booting up the system, loading necessary configurations, and initializing various agents within the system. This foundational step ensures that all agents are ready and equipped to perform assigned tasks.
- Assign Tasks to Agents 404: Following initialization, specific tasks are allocated to individual agents. Task assignment is typically based on the capabilities of each agent, ensuring that each part of the system is working on functions it is optimized for. This step might involve dynamic task allocation algorithms to match tasks with agents based on efficiency and capability.
- Coordinate Agent Interactions 406: Once tasks are assigned, this step manages the interactions between different agents. Effective coordination is crucial to prevent conflicts and ensure smooth operation. This includes synchronization of tasks, managing

dependencies, and handling communication between agents to align their activities towards common goals.

Feedback Collection and Processing 408: This step involves gathering feedback on the performance of agents and the overall system. Feedback can come from internal system assessments or external inputs. It is processed to evaluate the effectiveness of current strategies and agent performances.

Refine and Update Task Assignments 410: Based on the feedback received, this final step involves refining and updating the tasks assigned to agents. This iterative process ensures that the system continuously improves and adapts to new information or changes in the operating environment, optimizing task allocations to enhance overall system performance.

Overall, FIG. 4 illustrates a structured and cyclical process that allows a multi-agent AI system to operate efficiently, learn from interactions, and adapt to new conditions, thereby improving its decision-making capabilities continuously.

The following section outlines a broader Neuromorphic Architecture that encompasses COGNIGEN-AX, outlining a synergistic operation of its various components.

Neuromorphic architecture^[KX1]

COGNIGEN-AX is part of a narrative intelligence system based on an architectural pattern that draws inspiration from the structure and processes of the human brain and nervous system. This approach, known as neuromorphic or brain-inspired computing, aims to replicate the exceptional cognitive abilities of biological neural networks in handling complex information. To achieve this, the system combines large language models (LLMs), quantum computing, and an advanced neuro-symbolic knowledge representation framework, surpassing the limitations of classical von Neumann architectures when dealing with intricate, unstructured data

The components of this architectural pattern are as follows:

1. Neuromorphic Reasoning Core (COGNIGEN-AX): This component mimics the neural networks and cognitive processes of the brain, with a specific focus on modeling the role of the prefrontal cortex in metacognition and self-regulation. Much like the neural networks in the prefrontal cortex, COGNIGEN-AX continuously monitors and adjusts its knowledge models and reasoning strategies. It also integrates multimodal sensory data into coherent episodic narratives that capture contextualized meanings, similar to the brain's semantic memory.

2. Neuro-symbolic Knowledge Representation (GenFoundry): This component seamlessly combines the strengths of neural networks and symbolic reasoning for knowledge representation, thus emulating the brain's ability to encode and retrieve distributed conceptual representations. Acting as the foundation for COGNIGEN-AX, GenFoundry provides a neuro-symbolic knowledge base that is inspired by the brain's distributed conceptual representations. It encodes and retrieves contextualized knowledge fragments as high-dimensional vector embeddings, leveraging the distributional semantics and compositional abilities of LLMs. This integration allows for the capture of intricate relationships and affordances in the data, resembling the brain's ability to construct detailed situational models.

3. Quantum Neural Networks (Quantum Computing Optimization Framework): This component harnesses the potential of quantum computing by using quantum neural networks and tensor networks to process narratives in a quantum-enhanced format. To overcome the limitations of classical computing, the Quantum Computing Optimization Framework employs quantum neural networks and tensor networks to process the narratives generated by COGNIGEN-AX in a quantum-enhanced format. Building on quantum cognition theories, this framework utilizes quantum parallelism and superposition to explore multiple interpretive pathways simultaneously. By doing so, it uncovers deeper semantic associations and inferential links that classical methods cannot detect. The integration of these biomimetic components creates a narrative intelligence system that emulates the brain's ability to construct coherent narratives from fragmented sensory inputs. It addresses the challenges identified by Carbone and Mauch by combining neuromorphic reasoning, neuro-symbolic knowledge representation, and quantum cognitive computing.

The integration of these biomimetic components creates a narrative intelligence system that emulates the brain's ability to construct coherent, contextually grounded narratives from fragmented sensory inputs. By combining neuromorphic reasoning, neuro-symbolic knowledge representation, and quantum cognitive computing, this system addresses the following challenges:

1. Neural Information Processing: Similar to the brain's parallel distributed processing, the system uses quantum computing to process vast amounts of unstructured data at incredible speeds, overcoming the limitations of classical architectures.

2. Semantic Encoding: GenFoundry's neuro-symbolic representations capture the rich meanings, relationships, and affordances of the data, reflecting the brain's neural coding of conceptual knowledge. This provides a structured framework for encoding complex information.

3. Episodic Narrative Construction: COGNIGEN-AX emulates the brain's construction of coherent episodic memories from multimodal sensory inputs. It weaves narratives that connect different data elements, revealing deep insights similar to the brain's semantic memory retrieval.

- 5 4. Metacognitive Adaptation: COGNIGEN-AX continuously refines its knowledge models and reasoning processes, modeling the brain's self-reflective monitoring and adaptation of cognitive strategies. This enables contextually grounded decision-making.

This biomimetic narrative intelligence system presents a plausible model for encoding, constructing, and leveraging narratives to understand complex data landscapes. It
10 seamlessly integrates neural computation, symbolic knowledge representation, and quantum cognitive processing. Based on ontological principles and neural science, it paves the way for artificial general intelligence that can fluidly reason, adapt, and make decisions in real-world complexity.

Agent: MOPN

- 15 Modular and Hierarchical Multi-Agent Orchestration using Petri Nets (MOPN) Agent Methods

The MOPN agent is a modular and hierarchical multi-agent orchestration engine that uses executable Petri Net models. These models have formalized interactions and clear state specifications to control the execution flow and state transitions of different agents. Each agent's definitions and behavior are stored within reusable and versionable Petri Net
20 modules, which are maintained under strict version control using industry-standard branching and merging practices.

Agent Definition and Modeling:

MOPN defines each agent as a separate Petri Net module, encapsulating its specific functionality and behavior within a well-defined structure.

- 25 MOPN employs the following techniques for agent definition and modeling:

Petri Net Formalism:

Places: Represent the states or conditions of an agent, such as "Idle," "Processing," or "Completed."

Transitions: Represent the actions or events that trigger state changes in an agent, such as
30 "Start Processing" or "Finish Processing."

Arcs: Connect places and transitions, specifying the flow of tokens and the enabling conditions for transitions.

Hierarchical Decomposition:

5 Subnet Abstraction: Allows complex agent behaviors to be modeled as hierarchical subnets, encapsulating subprocesses and interactions within a higher-level net.

Subnet Invocation: Enables the invocation of subnets from higher-level nets, facilitating the modularization and reuse of agent behaviors.

State Specification:

10 Initial Marking: Defines the initial distribution of tokens in the places of an agent's Petri Net, representing its starting state.

Reachability Analysis: Determines the set of all possible states an agent can reach from its initial marking, ensuring the completeness and correctness of the model.

Version Control:

15 Branching: Allows parallel development and experimentation with different versions of agent models using techniques like feature branches or release branches.

Merging: Facilitates the integration of changes and updates to agent models using techniques like pull requests or merge requests, ensuring consistency and maintainability.

Interaction Formalization:

20 MOPN formalizes the interactions between agents using well-defined interfaces and communication protocols, ensuring an unambiguous and reliable exchange of information.

MOPN employs the following techniques for interaction formalization:

Interface Definition:

Input Places: Specify the expected inputs or preconditions for an agent to perform its functionality, represented as places in the agent's Petri Net.

25 Output Places: Specify the produced outputs or postconditions of an agent's functionality, represented as places in the agent's Petri Net.

Message Passing:

Asynchronous Communication: This enables agents to communicate asynchronously by passing messages through shared places or channels, which decouples the sender and receiver agents.

5 Synchronous Communication: This allows agents to communicate synchronously by enabling transitions simultaneously in the sender and receiver agents' Petri Nets.

Token-based Data Exchange:

Colored Tokens: These represent complex data structures or objects that are exchanged between agents. They are associated with specific token types or colors in the Petri Net.

10 Token Matching: This ensures that the exchanged tokens match the expected input and output types of the agents, preventing type mismatches or incompatible data exchanges.

Execution Flow and Orchestration:

MOPN controls the execution flow and orchestration of agents using a top-level Petri Net. This net defines the sequence and synchronization of agent invocations.

MOPN employs the following techniques for execution flow and orchestration:

15 Top-level Petri Net:

Agent Places: These represent the agents involved in the orchestration, with each agent modeled as a separate place in the top-level net.

Orchestration Transitions: These represent the control flow and synchronization points between agents. They specify the order and conditions for agent invocations.

20 Parallel Execution:

Concurrency: This allows multiple agents to execute concurrently by enabling transitions in their respective Petri Nets simultaneously.

Synchronization: This ensures proper synchronization between concurrently executing agents using synchronization places or transitions in the top-level net.

25 Hierarchical Orchestration:

Subnet Orchestration: This enables the orchestration of hierarchical subnets within agents, controlling the execution flow and synchronization at different levels of abstraction.

Recursive Invocation: This allows the recursive invocation of agents or subnets, enabling complex and iterative processing patterns.

Timed Transitions:

- 5 Delay Specification: This associates time delays with transitions in the Petri Net, representing the expected duration of agent processing or communication.

Timeout Handling: This incorporates timeout mechanisms to handle cases where agents exceed the specified time delays, ensuring the overall progress and reliability of the orchestration.

- 10 Petri Net Analysis and Verification:

MOPN incorporates rigorous analysis and verification techniques to ensure the correctness, reliability, and performance of the multi-agent orchestration.

MOPN employs the following techniques for Petri Net analysis and verification:

Reachability Analysis:

- 15 State Space Exploration: This generates the reachability graph or state space of the Petri Net, representing all possible markings reachable from the initial marking.

Boundedness Verification: This verifies that the number of tokens in each place remains bounded, preventing overflows or unbounded accumulation of tokens.

Liveness Analysis: Proofread this for improving clarity and flow:

- 20 Deadlock Detection: Identifies deadlock states in the Petri Net where no transition is enabled, indicating a halt in the system's progress.

Livelock Detection: Detects livelock situations where the system continuously transitions between states without making effective progress.

Performance Analysis:

- 25 Throughput Estimation: Estimates the maximum throughput or processing rate of the multi-agent system based on the Petri Net structure and transition firing rates.

Bottleneck Identification: Identifies potential bottlenecks or resource contention points in the Petri Net, helping to optimize the system's performance and resource allocation.

Monitoring and Logging:

MOPN incorporates comprehensive monitoring and logging mechanisms to track the execution and performance of the multi-agent orchestration.

MOPN employs the following techniques for monitoring and logging:

Instrumentation:

- 5 Place Marking Monitoring: Records the marking of places throughout the execution, capturing the state changes and token distributions in the Petri Net.

Transition Firing Monitoring: Logs the firing of transitions, including the timestamp, source, and target places, and associated data or events.

Event Logging:

- 10 Structured Logging: Generates structured log entries for each significant event or state change in the multi-agent system, following a consistent format.

Log Aggregation: Collects and aggregates log entries from different agents or components, providing a centralized view of the system's behavior.

Performance Metrics:

- 15 Execution Time Tracking: Measures the execution time of individual agents, transitions, or subnets, helping to identify performance bottlenecks or inefficiencies.

Resource Utilization Monitoring: Monitors the utilization of computational resources, such as CPU, memory, or network bandwidth, to optimize the system's performance and scalability.

Anomaly Detection:

- 20 Log Pattern Analysis: This technique applies pattern recognition and anomaly detection techniques to identify unusual or unexpected behaviors in log data.

Alert Generation: This function generates alerts or notifications when predefined thresholds or anomalous patterns are detected, enabling proactive issue resolution and system maintenance.

- 25 Output Format and Persistence:

MOPN supports standardized output formats and persistence mechanisms to facilitate data exchange, analysis, and integration with other systems.

MOPN employs the following techniques for output format and persistence:

ISO/IEC 15909 Compliance:

Part 1 (P/T Nets): Adheres to the ISO/IEC 15909-1 standard for the representation and exchange of Petri Nets, ensuring compatibility and interoperability with other tools and systems.

- 5 Part 2 (High-level Petri Nets): Supports the ISO/IEC 15909-2 standard for the representation and exchange of high-level Petri Nets, enabling the modeling of complex data types and behaviors - JSON Serialization:

Net Structure Serialization: Serializes the structure of the Petri Net, including places, transitions, and arcs, into a JSON format for storage and transmission.

- 10 Marking Serialization: Serializes the marking of the Petri Net, which represents the distribution of tokens across places, into a JSON format.

Database Persistence:

Relational Database: Stores the Petri Net structure, marking, and execution history in a relational database, allowing for efficient querying and analysis.

- 15 NoSQL Database: Utilizes a NoSQL database, such as MongoDB or Cassandra, to store and retrieve large-scale Petri Net data, providing scalability and flexibility.

Version Control Integration:

Git Integration: Integrates with version control systems like Git to track changes and versions of the Petri Net models and configurations.

- 20 Versioning Metadata: Includes versioning metadata, such as commit hashes or tags, in the serialized output to enable traceability and reproducibility.

Petri Net Visualization and Debugging:

MOPN incorporates visualization and debugging tools to support the development, analysis, and troubleshooting of Petri Net models.

- 25 MOPN employs the following techniques for Petri Net visualization and debugging:

Graphical Representation:

Place-Transition Graphs: Visualizes the Petri Net as a graphical place-transition graph, with circles representing places, rectangles representing transitions, and arrows representing arcs.

Marking Visualization: Displays the current marking of the Petri Net, showing the distribution of tokens across places using colors, numbers, or other visual cues.

Interactive Simulation:

- 5 Step-by-step Execution: Allows users to simulate the execution of the Petri Net step by step, enabling the inspection of state changes and transition firings.

Breakpoint Setting: Supports the setting of breakpoints at specific transitions or places, enabling the suspension of execution for debugging and analysis.

Debugging Tools:

- 10 Token Tracing: Provides tools for tracing the flow of tokens through the Petri Net, helping to identify incorrect or unintended behavior.

Reachability Graph Visualization: Generates and visualizes the reachability graph of the Petri Net, facilitating the analysis of state spaces and potential issues.

Performance Profiling:

- 15 Execution Time Analysis: Measures and visualizes the execution time of individual transitions, places, or subnets, helping to identify performance bottlenecks.

Resource Utilization Profiling: Monitors and visualizes the utilization of computational resources during the execution of the Petri Net, assisting in performance optimization.

- 20 The COGNIGEN-AX system employs a modular and hierarchical multi-agent orchestration approach using Petri Net models to coordinate the various AI agents and components. This orchestration addresses potential conflicts, ensures proper synchronization, and defines clear roles for the agents within the Petri Net execution framework. Here are some key aspects:

Conflict Resolution:

- 25 Clear Interface Definitions: Each agent in COGNIGEN-AX has well-defined input and output interfaces specified as "places" in their respective Petri Net modules. This ensures that agents only communicate and exchange data through designated channels, minimizing direct conflicts.

Token-based Coordination: Agents interact by consuming and producing tokens that flow through the Petri Net. Conflicts over shared resources are avoided by enforcing mutual exclusion constraints - only one agent can consume a token at a time.

Prioritization Strategies: If multiple agents are eligible to consume the same token (potential conflict), prioritization strategies like static priorities or dynamic priorities based on agent states can be employed to resolve disputes deterministically.

Agent Synchronization:

Synchronous Communication: The Petri Net formalism allows for synchronous communication between agents by having transitions that are part of multiple agent nets, forcing the corresponding agents to synchronize when such transitions fire.

Rendezvous Synchronization: Dedicated synchronization places can be introduced in the top-level orchestration of Petri Net. These places act as rendezvous points where tokens from multiple agents must accumulate before proceeding, ensuring synchronization.

Semaphore-based Synchronization: Semaphore concepts can be modeled in Petri Nets using complementary places to control access to shared resources and enforce synchronization between relevant agents.

Roles of Agents in Petri Nets:

The MOPT (Multi-agent Orchestration through Petri Nets) agent defines and manages the top-level Petri Net that orchestrates the execution flow and interactions between other agents like REIMAS, IPASE, COGNATE, etc.

Specialized agents like MACQ, RAINS, MIRROR (under IPASE) and CODA, OMFI (under REIMAS) are encapsulated within separate Petri Net modules playing distinct roles:

MACQ conducts modularized analysis and quantification of output quality

RAINS performs root cause analysis for identified limitations

MIRROR enables textual self-reflection and improvement suggestions

CODA extracts concepts, analyzes outputs, and detects drift

OMFI integrates feedback from multiple sources

The LITE (Logging and Instrumentation) agent has a cross-cutting role. It instrumented the execution of all Petri Net modules to generate comprehensive logs for traceability and analysis.

5 The AIKA (Autonomous Input and Knowledge Analysis) agent acts as a precursor, optimizing the initial user prompts before executing the main agent workflows.

COGNATE techniques like verbalized confidence, token probabilities, etc., play roles in the main workflow for estimating and integrating confidence scores.

10 The Petri Net models provide a formal framework to define agent roles, interactions, and execution semantics while enabling techniques like interfaces, tokens, priorities, and semaphores to manage conflicts and synchronize the execution of multiple autonomous agents in the COGNIGEN-AX system.

Example Scenario:

The user provides the following input record:

15 SARB:ei93:RTG11,% Premium method ranking,ei93,RTG11, Hierarchy for ranking of methods used in provisions assumptions (enumeration),Ratings,,"Method no.1 in ranking,Method no.2 in ranking,Method no.3 in ranking,Method no.4 in ranking,Method no.5 in ranking,Method no.6 in ranking,Method no.7 in ranking,Method no.8 in ranking,Method no.9 in ranking"

20 The goal is to map this record to the specified metadata attributes and generate a comprehensive output.

Multi-Agent Orchestration:

The AIKA (Autonomous Input and Knowledge Analysis) agent receives the user input record, which optimizes the initial prompt by understanding the context, evaluating ethics, preserving core intent, and adding clarifying details.

25 The optimized prompt is then passed to the MOPT (Multi-agent Orchestration through Petri Nets) agent, which initiates the execution of the top-level Petri Net orchestrating the interactions between the main agents: REIMAS, IPASE, and COGNATE.

30 The LITE (Logging and Instrumentation) agent is triggered to start logging the execution details, including net transitions, token movements, agent inputs/outputs, and any errors that may occur during the process.

The REIMAS (Retrospective Experience-driven Iterative Multi-model Adaptive Self-optimization) agent is activated first. It delegates the concept extraction, output analysis, and drift detection tasks to the CODA agent and the feedback integration task to the OMFI agent.

5 The CODA agent extracts the key concepts from the input record, such as "% Premium method ranking," "Hierarchy for ranking of methods used in provisions assumptions," and "Ratings." It analyzes the output and checks for any potential drift. The results are passed back to the REIMAS agent.

10 The OMFI agent collects feedback from various sources, such as domain experts or historical data, regarding the "% Premium method ranking" concept and its related metadata attributes. This feedback is integrated and provided to the REIMAS agent.

The MOPT agent then triggers the IPASE (Introspective Performance Analysis and Self-Enhancement) agent. The IPASE agent coordinates the activities of the MACQ, RAINS, and MIRROR agents.

15 The MACQ agent conducts a modularized analysis of the initial output generated by the REIMAS agent. It evaluates the output's relevance, coherence, factual accuracy, and completeness. For example, it may assess the significance of the "% Premium method ranking" concept to the insurance provisioning process and the completeness of its description.

20 The RAINS agent performs a root cause analysis to identify any limitations or areas for improvement identified by the MACQ agent. It uses techniques like the 5 Whys analysis and Fishbone diagrams to determine the underlying factors contributing to the issues, such as lack of domain knowledge or insufficient contextual information.

25 The MIRROR agent generates a self-reflective critique of the output, highlighting strengths, weaknesses, and suggestions for improvement. It may identify that the description of the "% Premium method ranking" concept lacks details on its calculation and implications and suggests incorporating more contextual information.

The IPASE agent consolidates the findings from the MACQ, RAINS, and MIRROR agents and passes the results back to the MOPT agent.

30 The MOPT agent then activates the COGNATE (Composable Game-theoretic Nash-embedded Adaptive Techniques for Estimating Confidence) agent to estimate the confidence scores for the generated output using techniques like Verbalized Confidence, Token Probabilities, and Prompt Entropy.

The confidence scores from different techniques are combined using the Nash Embedding approach, which treats each technique as a self-interested agent in a non-cooperative game. The Nash equilibrium is computed to determine the optimal composition of confidence scores.

- 5 The MOPT agent receives the integrated confidence scores from the COGNATE agent and determines if the output meets the desired confidence threshold.

If the confidence is high, the MOPT agent generates the final output, including the mapped metadata attributes, descriptions, and confidence scores. If the confidence is low, the MOPT agent may trigger additional iterations of the REIMAS, IPASE, and COGNATE agents to
10 refine the output.

Throughout the execution, the LITE agent captures detailed logs of the agent interactions, token movements, and any errors. These logs are stored in a structured format for traceability and analysis.

- 15 Finally, the MOPT agent returns the generated output to the user, including confidence scores and relevant explanations.

Conflict Resolution and Synchronization:

- The MOPT agent's Petri Net model resolves agent conflicts through clear interface definitions and token-based coordination. For example, suppose the MACQ and RAINS agents attempt to access the same output data simultaneously. In that case, the Petri Net
20 enforces mutual exclusion, allowing only one agent to proceed.

- Synchronization between agents is achieved through rendezvous points in the Petri Net. For instance, the IPASE agent waits to complete the MACQ, RAINS, and MIRROR agents before consolidating their findings. This synchronization is modeled using dedicated places in the Petri Net where tokens from multiple agents must accumulate before the transition can
25 fire.

The LITE agent's logging functionality is synchronized with the execution of other agents through carefully placed instrumentation points in the Petri Net. This ensures the logs capture the precise sequence of agent interactions and state transitions.

- 30 Agent: LITE

Logging and Instrumentation for Traceability and Explainability (LITE) Agent Methods

The LITE agent is responsible for creating instrumented logs that capture the execution details of the multi-agent orchestration process. It follows a standardized logging format based on the ISO 11179 metadata standard, storing information about Petri Net transitions, token movements, agent inputs and outputs, and error states. The logs generated by LITE enable traceability, explainability, and analysis of the system's behavior.

Transition Sequence Logging:

LITE captures the ordered sequence of state changes within the Petri Net model during execution, providing a step-by-step record of the system's behavior.

LITE employs the following techniques for transition sequence logging:

10 Transition Firing Logging:

Transition Identifier: Records the unique identifier of each transition that fires during the execution, allowing for precise tracking of state changes.

Source and Target Places: Logs the source and target places associated with each transition firing, capturing the flow of tokens and the enabling conditions.

15 Timestamp Recording:

Absolute Timestamp: Captures the absolute timestamp of each transition firing, providing a temporal record of the system's execution.

Relative Timestamp: Records the relative timestamp of each transition firing concerning the start of the execution or a reference point, enabling duration analysis.

20 Transition Metadata:

Transition Name: Logs the human-readable name of each transition, enhancing the interpretability of the transition sequence log.

Transition Description: Captures a brief description or annotation of each transition, providing additional context about the purpose or functionality of the transition.

25 Token Movement Tracking:

LITE tracks the flow of tokens between places in the Petri Net, representing data or control flow within the multi-agent system.

LITE employs the following techniques for token movement tracking:

Token Instance Logging:

Token Identifier: Assigns a unique identifier to each token instance, enabling the tracking of individual tokens throughout the execution.

Source and Target Places: Records the source and target places for each token movement, capturing the flow of tokens between agents or components.

5 Token Timestamp:

Movement Timestamp: Logs the timestamp of each token movement, providing a temporal record of data or control flow within the system.

Token Payload Logging:

10 Data Element Recording: Captures the data elements associated with each token, such as input parameters, intermediate results, or output values.

Payload Serialization: Serializes the token payload using a standardized format, such as JSON or Protocol Buffers, ensuring consistency and interoperability.

Agent Input/Output Logging:

15 LITE records the data exchanged between agents via input and output tokens, conforming to the ISO 11179 metadata standard for structured representation and semantic interoperability.

LITE employs the following techniques for agent input/output logging:

Input Token Logging:

20 Token Identifier: Captures the unique identifier of each input token received by an agent, enabling the tracking of data flow and dependencies.

Metadata Recording: Logs the metadata associated with each input token, such as the concept name, data type, and semantic meaning, conforming to the ISO 11179 standard.

Output Token Logging:

25 Token Identifier: Records the unique identifier of each output token produced by an agent, facilitating the tracking of data flow and agent interactions.

Metadata Recording: Captures the metadata associated with each output token, such as the concept name, data type, and semantic meaning, adhering to the ISO 11179 standard.

Payload Persistence:

Structured Storage: Stores the input and output token payloads in a structured format, such as a relational database or a document-oriented database, enabling efficient querying and analysis.

5 Retrieval and Linkage: Provides mechanisms for retrieving and linking the input and output token payloads based on their identifiers and metadata, facilitating traceability and data lineage analysis.

Error and Exception Logging:

LITE captures any exceptions, faults, or failures that occur during the execution of the Petri Net or within individual agents, enabling error analysis and debugging.

10 LITE employs the following techniques for error and exception logging:

Error Identification:

Error Code: Assigns a unique error code to each exception or fault, providing a standardized way to identify and classify errors.

15 Error Description: Logs a human-readable description of the error, explaining the nature and context of the exception or fault.

Error Origin:

Agent Identifier: Captures the identifier of the agent where the error occurred, enabling the localization of issues and facilitating agent-specific debugging.

20 Transition or Place: Records the specific transition or place within the agent's Petri Net where the error occurred, providing fine-grained details for error analysis.

Error Metadata:

Timestamp: Logs the timestamp of the error occurrence, allowing for temporal analysis and correlation with other system events.

25 Stack Trace: Captures the stack trace or execution path leading to the error, facilitating root cause analysis and debugging.

Error Propagation:

Error Token Generation: Generates special error tokens in the Petri Net to propagate error information and trigger error handling or recovery mechanisms.

Error Propagation Path: Tracks the propagation path of error tokens through the Petri Net, enabling the analysis of error impact and system resilience.

Log Format and Structure: LITE follows a standardized logging format based on the ISO 11179 metadata standard, ensuring the consistency, interoperability, and semantic
5 meaningfulness of the logged information.

LITE employs the following techniques for log format and structure:

JSON Format:

Log Entry Structure: Represents each log entry as a JSON object, with predefined fields for different types of logged information (e.g., transitions, tokens, inputs/outputs, errors).

10 Field Naming Conventions: Follows consistent and descriptive field names based on the ISO 11179 metadata standard, promoting clarity and understandability.

Metadata Enrichment:

ISO 11179 Compliance: Includes metadata fields compliant with the ISO 11179 standard, such as concept names, data types, and semantic definitions, enhancing the semantic
15 interoperability of the logs.

Domain-Specific Metadata: Incorporates domain-specific metadata fields relevant to the multi-agent system, such as application context, business processes, or organizational roles.

Log Hierarchy and Linkage:

Parent-Child Relationships: Establishes parent-child relationships between log entries, representing the hierarchical structure of the Petri Net execution (e.g., subnet invocations, nested transitions).
20

Linkage Identifiers: Uses unique identifiers or references to link related log entries, such as input and output tokens, enabling seamless navigation and traceability across the log data.

Log Storage and Persistence:

25 LITE stores the generated logs in a structured and persistent manner, facilitating efficient retrieval, querying, and analysis of the logged information.

LITE employs the following techniques for log storage and persistence:

Database Storage:

Relational Database: Stores the log entries in a relational database, leveraging the structured nature of the JSON format and the ISO 11179 metadata standard.

NoSQL Database: Utilizes a NoSQL database, such as MongoDB or Elasticsearch, for scalable and flexible storage of large volumes of log data.

5 Indexing and Querying:

Indexing Strategies: Applies appropriate indexing strategies, such as B-trees or inverted indexes, to optimize the retrieval and querying of log entries based on specific fields or criteria.

10 Query Languages: Supports expressive query languages, such as SQL or MongoDB Query Language, to enable efficient and targeted retrieval of log data for analysis and visualization.

Data Retention and Archival:

Retention Policies: Implements data retention policies to govern the duration and criteria for retaining log data, balancing storage costs with the need for historical analysis and auditing.

15 Archival Mechanisms: Provides mechanisms for archiving older log data to secondary storage or long-term archival systems, ensuring data preservation while optimizing active storage utilization.

Log Analysis and Visualization:

Proofread lightly: LITE provides tools and techniques for analyzing and visualizing logged information, enabling insights into system behavior, performance, and reliability.

20 LITE employs the following techniques for log analysis and visualization:

Log Parsing and Extraction:

JSON Parsing: Utilizes JSON parsing libraries or tools to extract structured information from log entries, enabling programmatic processing and analysis.

25 Field Extraction: Extracts specific fields or subsets of logged information based on analysis requirements, such as filtering by agent identifier, time range, or error type.

Statistical Analysis:

Descriptive Statistics: Computes descriptive statistics, such as mean, median, and standard deviation, for relevant log fields, providing summaries and insights into system behavior.

Time Series Analysis: Applies time series analysis techniques, such as trend analysis,

seasonality detection, or anomaly detection, to identify patterns or deviations in logged data over time.

Data Visualization:

5 Charting and Graphing: Generates visual representations of logged data, such as line charts, bar graphs, or pie charts, to facilitate understanding and communication of system behavior.

Dashboard Creation: Builds interactive dashboards that combine multiple visualizations and filters, enabling users to explore and analyze logged information from different perspectives.

Anomaly Detection and Alerting:

10 Anomaly Detection Algorithms: Applies anomaly detection algorithms, such as clustering, outlier detection, or machine learning models, to identify unusual or unexpected patterns in logged data.

15 Alerting Mechanisms: Implements alerting mechanisms that trigger notifications or actions based on predefined anomaly thresholds or rules, enabling proactive monitoring and issue resolution.

Integration with Development Workflow:

LITE integrates with the development workflow, providing valuable information for debugging, profiling, and continuous improvement of the multi-agent system.

LITE employs the following techniques for integration with the development workflow:

20 Debugging and Troubleshooting:

Log-based Debugging: Utilizes logged information to aid in debugging and troubleshooting, enabling developers to trace the execution flow, identify error states, and diagnose issues.

25 Interactive Debugging: Integrates with interactive debugging tools, such as debuggers or IDEs, to provide log-based breakpoints, watchpoints, or conditional logging, facilitating targeted debugging sessions.

Performance Profiling:

Execution Time Analysis: Analyses logged timestamps to measure and profile the execution time of specific transitions, agents, or subnets, identifying performance bottlenecks or optimization opportunities.

Resource Utilization Monitoring: Correlates the logged information with resource utilization metrics, such as CPU usage or memory consumption, to assess the performance impact of different system components.

Continuous Improvement:

- 5 Log-driven Insights: Derives actionable insights from the logged information to guide system improvements, such as identifying frequently occurring errors, optimizing resource allocation, or refining agent interactions.

Feedback Loop: Establishes a feedback loop between the logged information and the development process, using insights gained from log analysis to enhance the multi-agent system's design and implementation iteratively.

10

Autonomous Input and Knowledge Analysis (AIKA) Agent

The AIKA agent is responsible for analyzing user prompts, understanding their context and needs, assessing ethical considerations, preserving the core intent, and enhancing the prompts for improved response generation. It employs advanced techniques in natural language processing, knowledge representation, and ethical reasoning to create optimized prompts that lead to more accurate and contextually relevant responses from the AI system.

15

Prompt Storage and Retrieval:

AIKA begins by storing the user's prompt in a structured format, allowing for efficient retrieval and analysis throughout the optimization process.

20

AIKA employs the following techniques for prompt storage and retrieval:

Prompt Representation:

Structured Format: Represents the user prompt using a structured format, such as JSON or XML, to capture its content, metadata, and any additional contextual information

Semantic Annotation: Annotates the prompt with semantic tags or labels to identify key entities, intents, and relationships within the text

25

Prompt Indexing:

Indexing Techniques: Applies indexing techniques, such as inverted indexing or full-text search indexing, to enable fast and efficient retrieval of prompts based on specific keywords or phrases

Metadata Indexing: Indexes the metadata associated with each prompt, such as timestamp, user ID, or domain tags, to facilitate prompt retrieval based on contextual factors

Prompt Storage:

5 Database Integration: Stores the prompts in a database system, such as a relational database or a NoSQL database, to ensure data persistence and scalability

Data Consistency: Implements data consistency mechanisms, such as transaction management or data validation, to maintain the integrity and reliability of the stored prompts

User Context Analysis:

10 AIKA analyzes the user's prompt to understand their level of expertise, specific needs, and contextual factors that influence the desired response.

AIKA employs the following techniques for user context analysis:

Natural Language Understanding (NLU):

15 Intent Recognition: Applies intent recognition algorithms, such as rule-based pattern matching or machine learning classifiers, to identify the primary intent or purpose behind the user's prompt

Entity Extraction: Utilizes named entity recognition (NER) techniques, such as conditional random fields (CRF) or recurrent neural networks (RNN), to extract relevant entities and their types from the prompt

Domain Knowledge Inference:

20 Knowledge Graphs: Leverages domain-specific knowledge graphs or ontologies to infer the user's domain of interest based on the entities and relationships present in the prompt

Semantic Similarity: Computes semantic similarity measures, such as cosine similarity or word embeddings, to match the prompt's content with predefined domain categories or taxonomies

25 User Profiling:

Expertise Level Estimation: Analyzes the complexity and specificity of the language used in the prompt to estimate the user's level of expertise in the relevant domain

Personalization Factors: Considers additional user-specific factors, such as previous interaction history or user preferences, to tailor the prompt optimization process

Ethical Considerations Assessment:

AIKA assesses the ethical implications of the user's prompt, identifying potential biases, harmful content, or unintended consequences that may arise from the generated response.

AIKA employs the following techniques for ethical considerations assessment:

5 Bias Detection:

Lexicon-based Approaches: Utilizes predefined lexicons or dictionaries of biased terms and phrases to identify potential biases in the user's prompt

Machine Learning Models: Applies machine learning models, such as logistic regression or support vector machines (SVM), trained on annotated datasets to detect biased language or stereotypes
10

Harmful Content Identification:

Content Filtering: Employs content filtering techniques, such as keyword matching or regular expressions, to identify explicit, offensive, or inappropriate content in the prompt

Sentiment Analysis: Uses sentiment analysis algorithms, such as rule-based approaches or deep learning models, to detect the emotional tone and polarity of the prompt
15

Ethical Reasoning:

Deontological Ethics: Applies deontological ethical principles, such as the categorical imperative or the principle of double effect, to evaluate the moral permissibility of the prompt and its potential responses

Consequentialist Ethics: Considers the potential consequences and outcomes of the generated response, weighing the benefits against the risks and harms to individuals or society
20

Prompt Reformulation:

Bias Mitigation: Suggests alternative wordings or rephrases the prompt to mitigate identified biases while preserving the core intent of the user
25

Content Moderation: Filters or replaces inappropriate or harmful content in the prompt with more neutral or acceptable alternatives

Core Intent Preservation:

AIKA identifies and preserves the fundamental purpose or goal behind the user's prompt, ensuring that the optimized prompt maintains alignment with the user's original intent.

AIKA employs the following techniques for core intent preservation:

Semantic Role Labeling (SRL):

- 5 Predicate-Argument Structure: Applies SRL techniques, such as PropBank or FrameNet, to identify the predicate (main verb) and its associated arguments (subject, object, etc.) in the prompt

Thematic Roles: Assigns thematic roles, such as agent, patient, or instrument, to the arguments of the predicate to capture the semantic relationships within the prompt

- 10 Discourse Analysis:

Rhetorical Structure Theory (RST): Utilizes RST to analyze the rhetorical relations between different segments of the prompt, such as elaboration, contrast, or cause-effect

Coherence Modeling: Builds coherence models, such as entity grids or centering theory, to ensure that the optimized prompt maintains the logical flow and coherence of the original

- 15 prompt

Intent Summarization:

Abstractive Summarization: Employs abstractive summarization techniques, such as sequence-to-sequence models or transformer-based architectures, to generate a concise summary of the core intent of the prompt

- 20 Extractive Summarization: Uses extractive summarization algorithms, such as TextRank or LexRank, to identify and extract the most relevant sentences or phrases that capture the essence of the user's intent

Prompt Enhancement:

- 25 AIKA enhances the user's prompt by adding relevant context, clarifying ambiguities, and incorporating domain-specific knowledge to improve the quality and specificity of the generated response.

AIKA employs the following techniques for prompt enhancement:

Knowledge Retrieval:

Information Retrieval: Utilizes information retrieval techniques, such as TF-IDF or BM25, to search and retrieve relevant knowledge snippets from external knowledge bases or corpora

Knowledge Graph Traversal: Traverses domain-specific knowledge graphs or ontologies to identify and extract contextually relevant entities, relationships, and attributes

5 Query Expansion:

Synonym Expansion: Expands the prompt with synonyms or related terms using lexical resources, such as WordNet or thesauri, to improve the coverage and diversity of the generated response

10 Hypernym/Hyponym Expansion: Incorporates hypernyms (more general terms) or hyponyms (more specific terms) of the entities in the prompt to provide additional context and specificity

Coreference Resolution:

Pronoun Resolution: Resolves pronouns in the prompt to their corresponding antecedents using rule-based or machine learning approaches, such as the Hobbs algorithm or neural coreference resolution models

15 Entity Linking: Links named entities in the prompt to their canonical references in knowledge bases or entity repositories to disambiguate and enrich the contextual information

Prompt Rewriting:

Syntactic Simplification: Rewrites complex or convoluted sentences in the prompt into more straightforward, more concise structures to improve clarity and comprehension

20 Paraphrasing: Generates alternative phrasings or rewordings of the prompt using paraphrasing models, such as encoder-decoder architectures or rule-based approaches, to enhance diversity and naturalness

Response Generation:

25 AIKA generates an initial response to the optimized prompt using the AI system's language model and knowledge retrieval components, considering the user's context and level of expertise.

AIKA employs the following techniques for response generation:

Language Modeling:

Neural Language Models: Utilizes state-of-the-art neural language models, such as GPT, BERT, or T5, to generate fluent and coherent responses based on the optimized prompt

Domain Adaptation: Fine-tunes the language model on domain-specific corpora or incorporates domain-specific vocabulary and patterns to improve the relevance and accuracy of the generated response

Knowledge Integration:

Knowledge-Grounded Generation: Integrates retrieved knowledge snippets or entities into the response generation process using techniques such as attention mechanisms or copy mechanisms

Knowledge Fusion: Combines information from multiple knowledge sources or modalities (e.g., text, images, tables) to enrich the generated response and provide a more comprehensive answer

Response Ranking:

Beam Search: Employs beam search algorithms to generate and rank multiple candidate responses based on their likelihood of relevance scores

Reranking Models: Applies reranking models, such as BERT-based classifiers or reinforcement learning agents, to select the most appropriate response from the generated candidates

Contextual Adaptation:

User Expertise Adaptation: Adjusts the complexity, technicality, and depth of the generated response based on the estimated expertise level of the user

Personalization: Incorporates user-specific preferences, interaction history, or demographic information to tailor the response to the individual user's needs and expectations

Metadata Generation and Storage:

AIKA generates and stores metadata associated with the prompt optimization process, including the original prompt, optimized prompt, user context information, and generated response, using a standardized format such as ISO 11179.

AIKA employs the following techniques for metadata generation and storage:

Metadata Schema:

ISO 11179 Compliance: Defines a metadata schema that adheres to the ISO 11179 standard for metadata registries, ensuring interoperability and standardization across different components and systems

Metadata Elements: Identifies the key metadata elements to be captured, such as prompt identifier, timestamp, user identifier, domain tags, and optimization parameters

Metadata Extraction:

Automatic Metadata Generation: Automatically extracts relevant metadata from the prompt, user context, and generated response using natural language processing techniques and predefined extraction rules

Manual Metadata Annotation: Provides interfaces or tools for manual annotation of metadata by domain experts or knowledge engineers to ensure the accuracy and completeness of the captured metadata

Metadata Storage:

Relational Databases: Stores the metadata in a relational database management system (RDBMS) using a normalized schema design to ensure data integrity and consistency

NoSQL Databases: Utilizes NoSQL databases, such as document-oriented or key-value stores, to store metadata in a flexible and scalable manner, accommodating varying metadata structures and volumes

Metadata Indexing and Retrieval:

Indexing Mechanisms: Implements indexing mechanisms, such as inverted indexes or B-trees, to enable fast and efficient retrieval of metadata based on specific criteria or keywords

Query Languages: Supports standard query languages, such as SQL or MongoDB Query Language, to facilitate complex metadata queries and aggregations.

COGNIGEN-AX has wide-reaching industrial applications in various sectors, offering opportunities for autonomous AI reasoners to enhance decision-making, problem-solving, and task execution. By integrating advanced metacognitive abilities and confidence estimation techniques, AI systems can improve efficiency, reliability, and transparency in AI-driven processes across industries.

Healthcare and Biomedical Research:

COGNIGEN-AX can be used to develop intelligent clinical decision support systems that assist healthcare professionals in diagnosing diseases, recommending treatments, and predicting patient outcomes. The self-reflective and self-optimizing capabilities of COGNIGEN-AX-based AI reasoners improve the accuracy and reliability of medical decision-making. Confidence estimation techniques provide clinicians with a clear understanding of the AI system's certainty in its recommendations. Additionally, COGNIGEN-AX can accelerate drug discovery, analyze complex biological data, and generate novel hypotheses for experimental validation in biomedical research.

Financial Services and Investment Management:

COGNIGEN-AX can create autonomous AI reasoners for financial market analysis, risk assessment, fraud detection, and investment portfolio optimization. The self-enhancing and self-adapting nature of COGNIGEN-AX-based AI systems enables continuous learning from market dynamics, adjustment of strategies based on evolving conditions, and provision of accurate financial insights. Confidence estimation techniques quantify the uncertainty associated with economic predictions, enabling informed decision-making and risk management.

Autonomous Vehicles and Transportation:

COGNIGEN-AX can be applied to develop self-driving cars, unmanned aerial vehicles (UAVs), and intelligent transportation systems. The metacognitive abilities of COGNIGEN-AX-based AI reasoners enable continuous monitoring and optimization of performance, adaptation to changing environments, and real-time self-aware decision-making. Confidence estimation techniques ensure the safety and reliability of autonomous vehicles by providing a measure of the AI system's certainty in its perception, planning, and control decisions.

Manufacturing and Industrial Automation: COGNIGEN-AX can be utilized in the development of intelligent manufacturing systems, predictive maintenance solutions, and supply chain optimization tools. These systems can continuously improve their efficiency, quality, and responsiveness to market demands through the self-reflective and self-optimizing capabilities of COGNIGEN-AX-based AI reasoners. Additionally, confidence estimation techniques can identify and address potential issues or anomalies in the manufacturing process for proactive maintenance and quality control interventions.

In the field of education and personalized learning, COGNIGEN-AX can be applied to create intelligent tutoring systems and adaptive learning platforms. These systems can provide personalized educational experiences by understanding the strengths, weaknesses, and

learning preferences of individual learners through the metacognitive abilities of COGNIGEN-AX-based AI reasoners. By dynamically adjusting instructional content and strategies, these systems can optimize learning outcomes. Confidence estimation techniques can gauge the AI system's certainty in assessing student understanding, offering targeted feedback and recommendations for improvement.

COGNIGEN-AX can also be employed in natural language processing and conversational AI applications. Advanced chatbots, virtual assistants, and dialogue systems can engage users in more natural and context-aware conversations. Through the self-reflective and self-optimizing capabilities of COGNIGEN-AX-based AI reasoners, these systems continuously learn from user interactions, adapt language models and knowledge bases, and provide accurate and relevant responses. Confidence estimation techniques ensure the reliability and appropriateness of the AI system's generated outputs, reducing the risk of misleading or irrelevant information.

In the field of cybersecurity and fraud detection, COGNIGEN-AX can be applied to develop intelligent solutions that autonomously identify and respond to threats and abnormal activities. The self-enhancing and self-adapting nature of COGNIGEN-AX-based AI reasoners enables these systems to learn from new attack patterns, optimize detection algorithms, and adapt to changing risk landscapes. Confidence estimation techniques quantify the AI system's threat assessments and fraud alerts, enabling effective prioritization of investigations and interventions by security analysts.

Finally, COGNIGEN-AX can be applied in environmental monitoring and sustainability efforts. COGNIGEN-AX offers a range of applications in various fields. In environmental monitoring, renewable energy management, and resource optimization, COGNIGEN-AX-based AI reasoners can analyze complex data, identify patterns, and make informed decisions to minimize ecological impact. Confidence estimation techniques further help in policy-making and sustainability planning.

In legal and compliance, COGNIGEN-AX can develop tools for legal research, contract review, and compliance monitoring. The AI reasoners continuously learn from legal precedents and regulations to provide accurate and up-to-date insights. Confidence estimation ensures reliable and unbiased legal analysis.

For creative industries and entertainment, COGNIGEN-AX can create AI-driven tools like music composition systems, video editing, and generative art platforms. The AI reasoners learn from user preferences to generate engaging content, with confidence estimation preventing low-quality or offensive material.

Beyond these examples, COGNIGEN-AX's principles can be adapted to various domains, where autonomous AI reasoners offer valuable insights and decision support. The self-enhancing and self-optimizing nature of COGNIGEN-AX-based AI, along with confidence estimation, make them suitable for applications requiring reliability, adaptability, and transparency. As industries increasingly embrace AI-driven solutions, the demand for advanced AI reasoners like COGNIGEN-AX is expected to grow.

While COGNIGEN-AX, the present invention, has been described with specific embodiments and examples, it is important to note that modifications, alterations, and adaptations can be made without straying from the scope and spirit of the invention. The following alternative embodiments and variations demonstrate the flexibility and extensibility of the COGNIGEN-AX system and methods:

1. Domain-specific instantiations: COGNIGEN-AX can be instantiated and optimized for specific domains like healthcare, finance, education, or entertainment. The system can be tailored to meet the unique requirements and constraints of each domain by adjusting the knowledge bases, reasoning strategies, and confidence estimation techniques. For example, in a healthcare setting, the system and methods could incorporate medical ontologies, clinical guidelines, and domain-specific evaluation metrics to ensure reliable and safe AI-generated recommendations.

2. Integration with other AI architectures: The modular design of COGNIGEN-AX allows for seamless integration with different AI architectures, such as deep learning models, knowledge graphs, or reinforcement learning agents. The system and methods can be adapted to work with existing AI solutions, enhancing them with metacognitive abilities and confidence estimation capabilities. For instance, COGNIGEN-AX could be integrated with a deep learning-based computer vision system to enable self-reflective analysis and confidence-aware decision-making in image recognition tasks.

3. Distributed and decentralized deployment: COGNIGEN-AX can be deployed in distributed and decentralized environments, like edge computing networks or blockchain-based systems. The coordination and synchronization of AI reasoners across multiple nodes or participants can be managed through multi-agent orchestration using Petri Nets. This variation facilitates the development of scalable, fault-tolerant, and privacy-preserving AI systems that leverage the collective intelligence of distributed agents.

4. Human-in-the-loop optimization: The ISOCLES workflow can be modified to incorporate greater human involvement in the optimization process. Human experts can provide direct guidance, feedback, and validation at various stages of the workflow, including setting

optimization objectives, reviewing intermediate results, or fine-tuning reasoning strategies. This variation allows for a more collaborative and interactive approach to AI development, ensuring alignment with human values and domain expertise.

Transfer learning and domain adaptation play a crucial role in enhancing the capabilities of the COGNIGEN-AX system and methods. This allows AI reasoners to leverage knowledge and skills acquired in one domain and apply them to related domains, resulting in improved performance. The REIMAS and IPASE modules can be extended to support cross-domain knowledge transfer, while the COGNATE module can incorporate domain adaptation techniques to estimate confidence. This variation enables the development of more versatile and efficient AI systems that can quickly adapt to new tasks and environments.

To further enhance the transparency and interpretability features of COGNIGEN-AX, advanced explainable AI techniques can be integrated. These include counterfactual explanations, concept activation vectors, or rule-based reasoning. Additionally, the system and methods can incorporate intuitive user interfaces and interaction mechanisms that allow users to explore, question, and provide feedback on the AI reasoner's outputs and explanations. This variation promotes better user understanding, trust, and acceptance of AI-generated results.

COGNIGEN-AX can also be extended to support continuous learning and adaptation based on real-time data streams and user interactions. By incorporating online learning algorithms and incremental model updates into the ISOCLES workflow, AI reasoners can continuously refine their knowledge and strategies. This variation enables the development of AI systems that can operate autonomously in evolving environments, adapting to changing user preferences, data distributions, and performance requirements.

To enable multi-modal and cross-modal reasoning, COGNIGEN-AX can be adapted to process and integrate information from various data modalities, such as text, images, audio, and sensor data. The REIMAS and IPASE modules can be extended to handle multi-modal inputs and outputs, while the COGNATE module can incorporate techniques for cross-modal confidence estimation. This variation allows for the development of more comprehensive and context-aware AI systems that can reason across different data representations and modalities.

Explicit ethical and responsible AI considerations can be integrated into the COGNIGEN-AX system and methods. The IPASE module can incorporate ethical reasoning frameworks, such as deontological or consequentialist approaches, to evaluate the moral permissibility and societal impact of the AI-generated outputs. The COGNATE module can include

fairness and bias detection techniques to ensure equitable treatment of different user groups. This variation promotes the development of AI systems that align with human values, mitigate potential risks and harms, and foster trust and accountability. Quantum computing integration: As quantum computing technologies become more advanced, COGNIGEN-AX can be enhanced to take advantage of the computational benefits of quantum processors. The REIMAS, IPASE, and COGNATE modules can be adapted to work with quantum algorithms for reasoning, optimization, and confidence estimation. The Petri Net-based orchestration can be expanded to coordinate both quantum and classical computing resources. This variation allows for the development of AI systems that can solve complex, high-dimensional problems and achieve significant performance improvements through quantum-enhanced processing.

These alternative embodiments and variations demonstrate the wide range of possibilities and applications for the COGNIGEN-AX system and methods. By customizing and extending the core components and techniques, COGNIGEN-AX can be tailored to address diverse challenges and requirements across different domains and deployment scenarios. The invention's modular and flexible design encourages continuous innovation and improvement, ensuring its relevance and impact as AI technologies and user needs evolve.

FIG. 5 depicts a schematic representation of computing device 500, which is configured to support the operations and functionalities of the COGNIGEN-AX system, including its neuromorphic architecture, as described in the embodiments herein. The device comprises a central processing unit (CPU) 522, which interfaces with various data storage and memory components: secondary storage 524, read-only memory (ROM) 526, and random-access memory (RAM) 528.

Within this configuration, a Quantum Computing Unit (QCU) 536 is integrated. The QCU 536 utilizes quantum mechanical phenomena to enhance computation. It is engineered to execute algorithms that are particularly suitable for quantum computation, such as those involving large-scale number factorization, quantum simulations, and specific optimization problems, enabling computational performance that exceeds the capabilities of a standalone CPU.

The secondary storage 524 may include a dedicated sector 524a containing instructions executable by both the CPU 522 and the QCU 536. These instructions enable the device 500 to perform operations that may be optimized through quantum computing.

The ROM 526 stores immutable code essential for an initial booting process and routine operations of the computing device 500. The RAM 528 provides volatile memory for immediate access to data by the CPU 522 and the QCU 536 during active tasks.

5 Peripheral devices are managed through input/output (I/O) interfaces 530, while network connectivity is enabled via a network interface 532. A graphics processing unit (GPU) 534 is present to handle parallel processing tasks, which may be separate from or integrated with quantum computing processes.

10 The CPU 522, as the primary processor for general computing tasks, is responsible for executing sequential operations and handling a variety of computational processes in the classical domain. It manages routine tasks with high efficiency and interfaces with the system's memory, including RAM 528 and ROM 526, for data storage and retrieval.

15 Within this dual-capability system, the CPU 522 often acts as a coordinator, determining when to engage the QCU 536 based on the computational requirements. For quantum-suitable tasks, the CPU 522 prepares and relays data to the QCU 536, which then processes this information utilizing its quantum computing power. After the QCU 536 completes the quantum processing, it can transmit the results back to the CPU 522, which may perform additional classical processing or output the results.

20 The collaboration between the CPU 522 and the QCU 536 effectively expands the device's 500 computational range, enabling it to switch between classical and quantum operations. This ensures that the device 600 utilizes the most efficient processing method available, whether that be the classical computation provided by the CPU 522 or the quantum processing offered by the QCU 536.

25 Although the computing device 500 is described with reference to a single computer, it should be appreciated that the computing device may be formed by two or more computers in communication with each other that collaborate to perform a task. For example, but not by way of limitation, an application may be partitioned in such a way as to permit concurrent and/or parallel processing of the instructions of the application. Alternatively, the data processed by the application may be partitioned in such a way as to permit concurrent and/or parallel processing of different portions of a data set by the two or more computers. In an embodiment, virtualization software may be employed by the computing device 500 to provide the functionality of a number of servers that is not directly bound to the number of computers in the computing device 500. In an embodiment, the functionality disclosed above may be provided by executing the application and/or applications in a cloud computing environment. Cloud computing may comprise providing computing services via a network connection using

dynamically scalable computing resources. A cloud computing environment may be established by an enterprise and/or may be hired on an as-needed basis from a third-party provider.

Additional components, such as one or more application specific integrated circuits, neuromorphic computing units, field programmable gate arrays, or other electronic or photonic processing components can also be included and used in conjunction with or in place of the processor 522 to perform processing operations. The processing operations can include machine learning operations, other operations supporting the machine learning operations, or a combination thereof.

The technical solution detailed in present disclosure may be embodied in the form of a computer program product. The computer program product may be stored in a non-volatile or non-transitory storage medium, which can be a compact disk read-only memory, USB flash disk, or a removable hard disk. The computer program product includes a number of instructions that enable a computer device (personal computer, server, or network device) to execute the methods provided in the embodiments described herein. For example, such an execution may correspond to a simulation of the logical operations, including the training and aggregation of model updates in the federated learning process, as described herein. The software product may additionally or alternatively include number of instructions that enable the computing device 600 to execute operations for configuring or programming a digital logic apparatus in accordance with embodiments of the present invention.

By programming and/or loading executable instructions onto the computing device, at least one of the CPU 522, the RAM 528, and the ROM 526 are changed, transforming the computing device in part into a specific purpose machine or apparatus having the novel functionality taught by the present disclosure. It is fundamental to the electrical engineering and software engineering arts that functionality that can be implemented by loading executable software into a computer can be converted to a hardware implementation by well-known design rules.

It will be appreciated to those skilled in the art that the preceding examples and embodiments are exemplary and not limiting to the scope of the present disclosure. Whilst the foregoing description has described exemplary embodiments, it will be understood by those skilled in the art that many variations of the embodiment can be made within the scope and spirit of the present invention. It shall be noted that elements of any claims may be arranged differently including have multiple dependencies, configurations and combinations.

Claims:

1. A method for generating a confidence score specific to an output from an artificial intelligence system comprising:

processing an input query by a language model to obtain an initial output;

5 computing a plurality of confidence estimates for the initial output, wherein each confidence estimate is computed based on a distinct computational technique; and

aggregating the confidence estimates based on a game-theoretical approach to generate an aggregated confidence score for the initial output.

10 2. The method of claim 1, further comprising applying the aggregated confidence score to adjust parameters of an artificial intelligence system.

3. The method of claim 2, wherein the adjusted parameters of the artificial intelligence system are based on model performance metrics corresponding to a level of confidence determined by the aggregated confidence score.

15 4. The method of any one of the preceding claims, wherein the game-theoretic approach is derived from a Nash Embedding Theorem, which optimizes the aggregation of the confidence estimates by evaluating an interdependency among the plurality of confidence estimates.

20 5. The method of any one of the preceding claims, wherein the distinct computational technique is selected from a group comprising: verbalized confidence assessments, token probability analysis, prompt entropy measurements, semantic output clustering, self-consistency diagnostics; and hidden state divergence metrics.

6. The method of any one of the preceding claims, further comprising using a structured data format to encode the confidence estimates and the aggregated confidence score, wherein the format is compatible with a standardized metadata representation.

25 7. The method of any one of the preceding claims, further comprising refining the game-theoretical approach based on feedback derived from system performance to enhance future confidence estimations.

30 8. The method of any one of the preceding claims, wherein the distinct computational technique includes calculating token probabilities by extracting SoftMax probabilities for each token from an output distribution of the language model and calculating an overall sequence probability as a product of the token probabilities.

9. The method of any one of the preceding claims, wherein the distinct computational technique includes determining a prompt entropy by prompting the AI system with multiple label options, calculating an entropy of a resulting probability distribution corresponding to the label options, and mapping entropy values to the confidence estimates based on a predetermined function.
10. The method of any one of the preceding claims, wherein the distinct computational technique includes analyzing a verbalized confidence by pattern matching to identify statements of confidence in an output of an AI system and assigning numerical scores to the identified statements based on a predefined mapping.
11. The method of any one of the preceding claims, wherein the distinct computational technique includes clustering semantic outputs by generating a plurality of output sequences, computing semantic embeddings for the output sequences, and using metrics to quantify a clustering quality.
12. A system for information processing, comprising:
- a data acquisition module configured to receive feedback corresponding to one or more outputs generated by a language model;
 - a cognitive reasoning module configured to evaluate the reasoning process of the language model, emulate cognitive functions including metacognitive processes, and generate an assessment based on an analysis of the received feedback, wherein the assessment includes classifying the one or more outputs into components, assigning quality scores for each component, and identifying an improvement corresponding to the one or more outputs;
 - a process adjustment module coupled to the cognitive reasoning module, configured to adjust the reasoning process of the language model based on the assessment; and
 - a refinement module coupled to the process adjustment module, configured to iteratively refine the reasoning process based on subsequent updates to the generated assessment until a performance threshold is met.
13. A method for information processing, comprising:
- receiving feedback corresponding to one or more outputs generated by a language model;
 - using a reasoning module to evaluate the reasoning process of the language model, emulate cognitive functions including metacognitive processes, and generate an assessment

based on an analysis of the received feedback, wherein the assessment includes classifying the one or more outputs into components, assigning quality scores for each component, and identifying an improvement corresponding to the one or more outputs;

adjusting the reasoning process of the language model based on the assessment;

5 and

iteratively refining the reasoning process based on subsequent updates to the generated assessment until a performance threshold is met.

14. The method of claim 13, wherein the classifying of the one or more outputs into components includes applying a modularized analysis that quantifies evaluation metrics for
10 each component based on relevance, coherence, factual accuracy, and completeness

15. The method of claim 13 or 14, wherein the assigning of quality scores for each component utilizes techniques such as token probabilities, prompt entropy, and semantic output clustering to calibrate the confidence scoring.

16. The method of any one of claims 13-15, wherein the identifying of an improvement
15 includes employing a counterfactual reasoning process to evaluate potential alternative outcomes and their impacts on the reasoning process of the language model.

17. The method of any one of claims 13-16, wherein the reasoning module further comprises a sub-module for generating detailed logs of each iteration in the refinement process, which includes recording changes to reasoning strategies and their effects on output quality.

20 18. A system for coordinating operations, comprising:

a model configuration module configured to define a structured framework of a multi-agent system, wherein each component of the multi-agent system is assigned tasks related to processing outputs generated by an artificial intelligence (AI) system; and

25 an agent coordination module configured to manage interactions and synchronize data flow among agents of the system based on roles or dependencies corresponding to each agent within the structured framework and utilize mechanisms for coordination and synchronization of the agents of the system.

19. The system of claim 18, wherein the model configuration module is further configured to utilize Petri nets to define the structured framework, wherein the Petri nets specify the roles
30 or dependencies of each agent within the multi-agent system.

20. The system of claims 18 or 19, wherein the agent coordination module utilizes a conflict resolution strategy to manage data flow among the agents when multiple agents access data resources simultaneously.

5 21. The system of any one of claims 18-20, wherein each agent within the multi-agent system is configured to generate and send feedback regarding their task execution to the model configuration module, which uses the feedback to refine task assignments in subsequent operations.

10 22. The system of any one of claims 18-21, wherein the model configuration module assigns the tasks to agents based on a dynamic assessment of the operational load and performance metrics of each agent.

23. A method for coordinating operations within a multi-agent system, comprising:

implementing a structured framework of a multi-agent system, wherein each component of the multi-agent system is assigned specific tasks related to processing outputs generated by an artificial intelligence (AI) system;

15 managing interactions and synchronizing data flow among agents of the multi-agent system based on roles or dependencies corresponding to each agent within the structured framework; and

utilizing mechanisms for coordination and synchronization of the agents of the system.

20

25