United States Patent Application Publication	20250259047
Kind Code	A1
Publication Date	August 14, 2025
Inventor(s)	Crabtree; Jason et al.

COMPUTING PLATFORM FOR NEURO-SYMBOLIC ARTIFICIAL INTELLIGENCE APPLICATIONS

Abstract

A distributed generative artificial intelligence (AI) reasoning and action platform that utilizes a cloud-based computing architecture for neuro-symbolic reasoning. The platform comprises systems for distributed computation, curation, marketplace integration, and context management. A distributed computational graph (DCG) orchestrates complex workflows for building and deploying generative AI models, incorporating expert judgment and external data sources. A context computing system aggregates contextual data, while a curation system provides curated responses from trained models. Marketplaces offer data, algorithms, and expert judgment for purchase or integration. The platform enables enterprises to construct user-defined workflows and incorporate trained models into their business processes, leveraging enterprise-specific knowledge. The platform facilitates flexible and scalable integration of machine learning models into software applications, supported by a dynamic and adaptive DCG architecture.

Inventors: **Crabtree; Jason (Vienna, VA), Kelley; Richard (Woodbridge, VA), Hopper; Jason (Halifax, CA), Park; David (Fairfax, VA)**

Applicant: **QOMPLX LLC** (Reston, VA)

Family ID: **1000007908924**

Appl. No.: **18/656612**

Filed: **May 07, 2024**

Related U.S. Application Data

us-provisional-application US 63551328 20240208

Publication Classification

Int. Cl.: **G06N3/0475** (20230101)

U.S. Cl.:

CPC **G06N3/0475** (20230101);

Background/Summary

CROSS-REFERENCE TO RELATED APPLICATIONS [0001] Priority is claimed in the application data sheet to the following patents or patent applications, each of which is expressly incorporated herein by reference in its entirety: [0002] 63/551,328

BACKGROUND OF THE INVENTION
Field of the Art
[0003] The present invention is in the field of large-scale cloud computing, and more particularly to distributed, graph-based computing platforms for artificial intelligence based decision-making and automation systems including those employing large language models (LLMs) and associated services across heterogeneous cloud, managed data center, edge, and wearable/mobile devices.
Discussion of the State of the Art
[0004] The recent rapid rise in the capabilities and uses of machine learning, artificial intelligence (AI) and more recently large language models (LLMs) in support of generative artificial intelligence (GenAI or GAI) applications has raised myriad new challenges. New issues such as model hallucination and the protection of intellectual property rights, particularly copyrights and trademarks, against unattributed and unauthorized exploitation by AI models including LLMs, have arisen, while well-known challenges such as user privacy and cybersecurity have become even more difficult to manage and understand. As importantly, dramatic new opportunities have arisen, as myriad participants rush to adopt more diverse data sources and feed growing machine learning, artificial intelligence and in particular GenAI and LLMs into their existing businesses and entirely new business categories.
[0005] The 2023/2024 AI boom has been largely focused on the Transformer and Diffusion models. Transformer models, which generate their output one piece at a time, use each newly generated fragment as part of the inputs for the next. The popular ChatGPT, Gemini, Bard, LLaMa, and GPT-4 are good examples of models that rely on this core approach. Closely related are the Diffusion models, which are used primarily to generate new images by beginning with noise and improving the quality across the entire image until prompts and imagery generate sufficient similarity; Dall-E, Midjourney, and Stable Diffusion are examples of this type of model.
[0006] It is important to note that foundational Large Language Models, like GPT-3 and GPT-4 and its variants, are considered connectionist AI models rather than symbolic AI. They each have a Connectionist Architecture: the currently in vogue LLMs are a special case of neural network architectures, specifically deep neural networks with millions to billions of parameters. These networks consist of interconnected nodes (neurons) organized in layers. The connections between neurons are weighted, and information flows through these connections during computation. These layers include an input layer, one or more hidden layers, and an output layer. This connectionist architecture is fundamentally different from the symbolic AI approach, which relies on explicit representations of symbols and rules. Further, they typically have a Distributed Representation: LLMs represent information in a distributed manner. Each neuron in the network contributes to the representation of multiple features or concepts simultaneously. In contrast, symbolic AI systems represent knowledge using discrete symbols (e.g. for objects, properties, and relationships) and rules for manipulating these symbols to enable reasoning and problem-solving. LLMs, by distributing information across a vast number of neurons and weights, can capture complex and nuanced patterns in data. These models all use Learning from Data: LLMs are trained through a

data-driven approach, where they learn patterns and relationships from massive amounts of data. Large language models like GPT (Generative Pre-trained Transformer) are typically trained on vast amounts of text data. However, there are advanced models that have been designed to work with different types of data including images, video, audio, tabular/structured, multimodal, code, DNA, and molecules, often building on top of foundational models trained on large text-dominant corpora. These models do not rely on predefined symbolic representations or explicit rules. Instead, they adjust their internal connections (weights) through training to improve their ability to predict the next element in a sequence, such as a word in a sentence, or generate coherent content. Symbolic AI, on the other hand, requires express encoding of knowledge and rules and in earlier eras this relied heavily on manually curated expert knowledge. Further, Connectionist models like LLMs are known for their ability to generalize from the data they have seen during training to generate novel and contextually relevant responses or attempts. They can handle a wide range of tasks and domains by leveraging learned representations that often provide surprisingly convincing or plausible results. Symbolic artificial intelligence (AI) systems often struggle with generalization, as they typically require explicit rules for each specific task or domain. Connectionist models like LLMs lack explicit symbolic rules; that is, they do not have explicit symbols, symbolic representations or rules encoded within them. They generate responses based on patterns and associations learned from data utilized in training processes, but in doing so are largely “black box”. In contrast, symbolic AI systems rely heavily on symbolic representations and rules that were once manually created and maintained and can be explained and traced as a result.

[0007] Overall, the connectionist nature of many current techniques like LLMs allows them to excel in various natural language understanding tasks and handle complex, context-rich data, making them a powerful tool for many potential AI applications such as content generation, chatbots, language translation, sentiment analysis, text summarization, classification and labeling, question answering and support applications, personalized or contextual recommendations, and comparative analysis. Symbolic AI, on the other hand, is traditionally viewed as being reliant on predefined symbols and rules. This requires formal treatment of data for analysis to enable association between the ontology associated with a given Symbolic application and a data set of interest. This need for highly congruent input and narrow domain framing and brittleness, which can be rigid and less adaptive in the face of real-world data variability, heterogencity, and uncertainty, can limit its practical applications in many ways. But this common framing of Connectionist versus Symbolic fails to account for broader “systems” level views of practical AI applications which are ultimately of interest to prospective human or robotic agents, owners, and supervisors—who ultimately wish to accomplish something better, cheaper, and/or faster than would otherwise be possible with traditional data analytics, machine learning, or AI systems of either symbolic or connectionist origin.

[0008] Within the LLM space using transformer-based models, word embedding is a technique that assigns words to numerical vectors in a way that connects similar words with vectors that are close in proximity and separates dissimilar words with distant vectors. Similarly, sentence embedding associates a vector with each sentence, gauging the similarity between sentences by allocating large numbers to similar ones and small numbers to dissimilar ones. However, word embeddings have a significant limitation when dealing with words that have multiple meanings. For instance, if a word embedding assigns a vector to the word ‘bear,’ it assigns the same vector to all of its various definitions. This poses a challenge when you want to use the word ‘bear’ in different contexts. This is where the concept of attention comes into play.

[0009] Attention is a mechanism for distinguishing between words when they are used in diverse contexts, transforming word embeddings into contextualized word embeddings. The computer considers all the words in a sentence as context, even including seemingly irrelevant words like “the,” “of,” and “in.” However, it weighs them based on their similarity to the word ‘bear.’ In a well-designed embedding, the similarity between ‘bear’ and words like ‘the’ is nearly zero,

indicating their lack of relevance. Consequently, the model learns to ignore these insignificant words and focuses on those with higher similarity to 'bear.' Multi-head attention is a method that allows one to modify embeddings to create various attention mechanisms. These modifications can be trained, much as a neural network is trained to more precisely employ weightings reflecting multiple potential contexts-"Bear at the Zoo" is different from "Bear!" during a hiking trip is different from "bear with me" (i.e. be patient with me).

[0010] One of the overarching limitations about non-symbolic or Connectionist AI systems like LLMs is that they hallucinate, which can lead to dubious or truly false results. In some cases, LLMs have been known to make up and state facts that are untrue and were never part of their training data. As real-world consequences of making decisions or taking actions based on LLM outputs begin to appear—with concomitant safety and legal risks—connecting the output from a given LLM prompt to a knowledge base imbued with actual semantic meaning is needed. While many LLM companies like OpenAI and Anthropic and Cohere market things like Search Ranking or Semantic Search improvements using LLM overlays on traditional database queries (either user-constructed or constructed as an initial prompt output itself), the results, rankings etc. are not in fact imbued with any "semantic" knowledge or real understanding in the Symbolic sense. Such claims of semantic search are predicated on these kinds of similarity measures which indicate closeness based on contextual indicators derived from the training data—this is distinct from knowledge, comprehension, application, analysis, synthesis, and evaluation in the way used in Bloom's 1956 taxonomy or the 2001 revision which used Remember, Understand, Apply, Analyze. Evaluate, and Create. The effectiveness of meaning representation models or embeddings relies on distance measurement based on corpora of training data from which vector similarity is determined. The most popular vector similarity functions are Euclidean distance, cosine similarity, and the inner product. The most common linguistic structures in today's natural language tools remain bag of words (no structure), sequential, constituent, and dependency parsing.

[0011] While Connectionist outputs are "made up" they are often nevertheless very compelling artifices which generate believable output that is sometimes correct. Believability may, but does not necessarily, equate to correctness, usefulness, or more broadly fitness for purpose (or optimization) given practical, legal, ethical, moral, and economic considerations and constraints. This is further complicated by the need to update or evolve models, and the systems of which they are a part, on an ongoing basis whether through continuous learning processes with retraining, reinforcement learning, or techniques like partially neural reinforcement learning which provide frameworks for ongoing verification of neural network based models within a learning loop inside of continuous state and action spaces.

[0012] What is needed is a distributed graph-based computing platform for managing increasingly complex and heterogeneous machine learning and artificial intelligence enhanced data processing, that enables more flexible and contextual use of increasingly heterogeneous computing, transport, and storage technologies and balkanized technology, encryption, and privacy laws and regulations to support new business and technology opportunities to emerge at pace. Such a capability may also enable better responses to the new challenges raised by the dynamic new technology landscape faced by all after the rapid and haphazard introduction of myriad generative AI technologies and platforms, wearables and internet of things devices that are proliferating.

SUMMARY OF THE INVENTION

[0013] Accordingly, the inventor has conceived and reduced to practice, a distributed graph-based computing platform for operationalizing generative artificial intelligence including large language models and neuro-symbolic systems for just-in-time, just-in-context and just-in-place data transport, storage, and compute as part of decision-making and optimization within combined logical and physical processes. A distributed artificial intelligence (AI) reasoning and action platform that utilizes a cloud-based computing architecture for neuro-symbolic reasoning that can selectively and appropriately blend machine learning, artificial intelligence, statistics, and

simulation modeling processes in support of continuous learning from both empirical observations and hypothetical state space explorations of complex adaptive systems. The platform comprises systems for distributed computation, curation, marketplace integration, and context management support human-machine teaming within complicated and complex cyber physical environments that have a wide range of business and decision-making applications. A distributed computational graph (DCG) orchestrates complex workflows for building and deploying algorithms and models, incorporating expert judgment, and utilizing both internal and external data sources to improve a given system's, individual's, group's or organization's outcomes over time. This is facilitated in part by ongoing analysis of the system of interest to the observer, which is typically the beneficiary of the flow-based computing processes enabled by the system, and the evaluation of both experienced and hypothetical (e.g. via simulation or modeling) states over time under different perturbations or shocks from both internal or exogenous factors. A context computing system aggregates contextual data from local or global and internal or external sources, while a curation system provides curated responses from models (e.g. statistical, machine learning, simulation based, or artificial intelligence including generative approaches). Marketplaces offer data (including unstructured, structured, schematized, normalized, and semantified), algorithms, models, model components, worlds or artifacts of them, and expert judgment(s) for purchase or integration. The platform enables enterprises to construct user-defined workflows and incorporate trained models into their business processes, leveraging enterprise-specific knowledge. The platform facilitates flexible and scalable integration of statistical, machine learning and artificial intelligence and simulation models into software applications, supported by a dynamic and adaptive DCG architecture that supports execution of data flows and orchestration of resources across cloud (e.g. hyperscale), self-managed (e.g. traditional data center) compute clusters, CDNs (e.g. forward content distribution networks that may expand from historical distribution of web content into forward hosting of data sets, models, AI tools, etc. . . .), edge devices, wearables and mobile devices, and individual computers. This DCG architecture not only supports pipeline definition of transformation tasks, especially via a Data Service Layer (DSL) for recipe and template like creation of flows, but enables the express establishment of dependencies and bill of materials from resources, data sets, algorithms, models, and rules/regulations applicable to the data flow being established. This may also optionally include the consideration of both 1st, 3rd, and 4th party compute, transport and storage dependencies with optional declaration of compute transport or storage locality for express, resource-dependent or economically dependent, or allowed locality instruction. The inventions' ability to orchestrate just-in-time, just-in-place, and just-in-context data flow processing across ecosystems with multiple stakeholders who may have usage or licensing or economic considerations tied to highly distributed and heterogeneous data processing flows represents a substantial step forward above current cloud based orchestration of data flows and resources (including serverless kinds of architectures like Serverless Flink and schema registries) to provide a much more flexible, comprehensive and manageable means of enabling highly integrated multi-vendor/processor workflows for businesses, organizations, consumers, AI agents and automation engines, and programmers alike.

[0014] According to a preferred embodiment, a computing system for operationalizing generative artificial intelligence employing a distributed neuro-symbolic reasoning and action platform is disclosed, the computing system comprising: one or more hardware processors configured for: obtaining a plurality of input data, the input data comprising enterprise knowledge and expert knowledge; processing the obtained plurality of input data using an embedding model to create a vectorized dataset; training a machine learning model using the vectorized dataset as a training dataset, wherein the machine learning model is configured to learn representations of the vectorized dataset; mapping the learned representations to symbolic concepts or rules to create symbolic representations of the learned representations; applying symbolic reasoning techniques to the symbolic representations to perform a reasoning task; engineering a prompt using an output of the

reasoning task; submitting the prompt and a plurality of context elements to one or more large language models; curating one or more responses for the one or more large language models to create a single response or action; and transmitting the single response or action to a user device.
[0015] According to another preferred embodiment, a computer-implemented method executed on a distributed neuro-symbolic reasoning and action platform for operationalizing generative artificial intelligence is disclosed, the computer-implemented method comprising: obtaining a plurality of input data, the input data comprising enterprise knowledge and expert knowledge; processing the obtained plurality of input data using an embedding model to create a vectorized dataset; training a machine learning model using the vectorized dataset as a training dataset, wherein the machine learning model is configured to learn representations of the vectorized dataset; mapping the learned representations to symbolic concepts or rules to create symbolic representations of the learned representations; applying symbolic reasoning techniques to the symbolic representations to perform a reasoning task; engineering a prompt using an output of the reasoning task; submitting the prompt and a plurality of context elements to one or more large language models; curating one or more responses for the one or more large language models to create a single response or action; and transmitting the single response or action to a user device.
[0016] According to another preferred embodiment, a system for operationalizing generative artificial intelligence employing a distributed neuro-symbolic reasoning and action platform is disclosed, comprising one or more computers with executable instructions that, when executed, cause the system to: obtain a plurality of input data, the input data comprising enterprise knowledge and expert knowledge; process the obtained plurality of input data using an embedding model to create a vectorized dataset; train a machine learning model using the vectorized dataset as a training dataset, wherein the machine learning model is configured to learn representations of the vectorized dataset; map the learned representations to symbolic concepts or rules to create symbolic representations of the learned representations; apply symbolic reasoning techniques to the symbolic representations to perform a reasoning task; engineer a prompt using an output of the reasoning task; submit the prompt and a plurality of context elements to one or more large language models; curate one or more responses for the one or more large language models to create a single response or action; and transmit the single response or action to a user device.
[0017] According to another preferred embodiment, non-transitory, computer-readable storage media having computer executable instructions embodied thereon that, when executed by one or more processors of a computing system employing a distributed neuro-symbolic reasoning and action platform for operationalizing generative artificial intelligence, cause the computing system to: obtain a plurality of input data, the input data comprising enterprise knowledge and expert knowledge; process the obtained plurality of input data using an embedding model to create a vectorized dataset; train a machine learning model using the vectorized dataset as a training dataset, wherein the machine learning model is configured to learn representations of the vectorized dataset map the learned representations to symbolic concepts or rules to create symbolic representations of the learned representations; apply symbolic reasoning techniques to the symbolic representations to perform a reasoning task; engineer a prompt using an output of the reasoning task; submit the prompt and a plurality of context elements to one or more large language models; curate one or more responses for the one or more large language models to create a single response or action; and transmit the single response or action to a user device.
[0018] According to an aspect of an embodiment, the plurality of context elements comprises retrieval-augmented generation (RAG) information.
[0019] According to an aspect of an embodiment, the RAG is obtained from a RAG marketplace.
[0020] According to an aspect of an embodiment, the plurality of context elements comprises information associated with the user device.
[0021] According to an aspect of an embodiment, the plurality of context elements comprises information associated with an action a user of the user device is performing during interaction

with the platform.
[0022] According to an aspect of an embodiment, the expert knowledge comprises scored datasets or scored model output.
[0023] According to an aspect of an embodiment, the expert knowledge is obtained from an expert knowledge marketplace.

Description

BRIEF DESCRIPTION OF THE DRAWING FIGURES
[0024] FIG. **1** is a block diagram illustrating an exemplary system architecture for a distributed generative artificial intelligence reasoning and action platform, according to an embodiment.
[0025] FIG. **2** is a block diagram illustrating an exemplary aspect of a distributed generative AI reasoning and action platform incorporating various additional contextual data.
[0026] FIG. **3** is a diagram illustrating incorporating symbolic reasoning in support of LLM-based generative AI, according to an aspect of a neuro-symbolic generative AI reasoning and action platform.
[0027] FIG. **4** is a block diagram illustrating an exemplary architecture for a neuro-symbolic generative AI reasoning and action platform configured for federated learning at a plurality of edge devices, according to an embodiment.
[0028] FIG. **5** is a block diagram illustrating an exemplary architecture for a neuro-symbolic generative AI reasoning and action platform configured to utilize a midserver to act as a computing intermediary between a plurality of edge devices and the platform.
[0029] FIG. **6** is a block diagram illustrating an exemplary mobile device configured for experience curation using embedded capabilities and functionality provided by a neuro-symbolic generative AI reasoning and action platform, according to an embodiment.
[0030] FIG. **7** is a block diagram illustrating an exemplary aspect of a distributed generative artificial intelligence reasoning and action platform, a curation computing system.
[0031] FIG. **8** is a block diagram illustrating an exemplary aspect of a distributed generative artificial intelligence reasoning and action platform, a marketplace computing system.
[0032] FIG. **9** is a block diagram illustrating a simple example of a distributed computational graph representation for providing neuro-symbolic GenAI capabilities, according to an aspect.
[0033] FIG. **10** is a block diagram illustrating an exemplary aspect of an embodiment of a distributed computational graph computing system utilizing an advanced cyber decision platform (ACDP) for external network reconnaissance and contextual data collection.
[0034] FIG. **11** is a block diagram illustrating another exemplary aspect of an embodiment of a distributed computational graph computing systems utilizing an advanced cyber decision platform.
[0035] FIG. **12** is a diagram of an exemplary architecture for a system for rapid predictive analysis of very large data sets using an actor-driven distributed computational graph, according to one aspect.
[0036] FIG. **13** is a diagram of an exemplary architecture for a system for rapid predictive analysis of very large data sets using an actor-driven distributed computational graph, according to one aspect.
[0037] FIG. **14** is a diagram of an exemplary architecture for a system for rapid predictive analysis of very large data sets using an actor-driven distributed computational graph, according to one aspect.
[0038] FIG. **15** is a block diagram of an architecture for a transformation pipeline within a system for predictive analysis of very large data sets using distributed computational graph computing system.
[0039] FIG. **16** is a process flow diagram of a method for predictive analysis of very large data sets

using the distributed computational graph
[0040] FIG. **17** is a process flow diagram of a method for an aspect of modeling the transformation pipeline module as a directed graph using graph theory.
[0041] FIG. **18** is a flow diagram illustrating an exemplary method for providing experience curation, according to an aspect of an embodiment.
[0042] FIG. **19** is a flow diagram illustrating an exemplary method for providing experience curation with using rich contextual data, according to an aspect of an embodiment.
[0043] FIG. **20** is a flow diagram illustrating an exemplary method for providing distributed neuro symbolic reasoning and action, according to an aspect of an embodiment.
[0044] FIG. **21** is a flow diagram illustrating an exemplary method for using a distributed computation graph system for creating structured representations or knowledge graphs from various data sources, and setting up a pipeline for continuous processing and monitoring of that data, according to an embodiment.
[0045] FIG. **22** illustrates an exemplary computing environment on which an embodiment described herein may be implemented.

DETAILED DESCRIPTION OF THE INVENTION

[0046] A distributed graph-based computing platform for managing and operationalizing artificial intelligence enhanced decision-making and automation systems including large language models and neuro-symbolic systems. A distributed planning, machine learning, artificial intelligence, modeling simulation and generative artificial intelligence (AI) reasoning and action platform that utilizes a cloud-based computing architecture for operationalizing neuro-symbolic reasoning in real-world cyber physical applications. The platform comprises systems for distributed computation, curation, marketplace integration, and context management across heterogeneous computing environments across heterogeneous cloud, managed data center, edge, and wearable/mobile devices.
[0047] A distributed computational graph (DCG) creates, stores, analyzes, orchestrates, and refines complex workflows for building and deploying intelligent decision support, decision making, and automation systems for machine and human machine teamed processes leveraging, statistics, simulation modeling, machine learning, artificial intelligence, automated planning, and generative AI, incorporating expert judgment and internal and external data sources and marketplaces for data, models, algorithms, model weights, experts and third party APIs or services. A just-in-place, just-in-time, and just-in-context computing system aggregates data, while a curation system provides curated responses from selected trained models and resources. Marketplaces offer data, algorithms, databases, model weights and components, models and expert judgment for purchase or integration. The platform enables enterprises to construct user-defined workflows and incorporate trained models into their business processes, leveraging enterprise-specific knowledge. The platform facilitates flexible and scalable integration of machine learning models into software applications, supported by a dynamic and adaptive DCG architecture across heterogeneous resource pools with a declarative language domain-specific language for resource, transformation, and process flows across heterogeneous resource pools with awareness of legal, regulatory, privacy, economic, technology velocity (e.g. Flink or Kafka becoming more or less active).
[0048] One or more different aspects may be described in the present application. Further, for one or more of the aspects described herein, numerous alternative arrangements may be described; it should be appreciated that these are presented for illustrative purposes only and are not limiting of the aspects contained herein or the claims presented herein in any way. One or more of the arrangements may be widely applicable to numerous aspects, as may be readily apparent from the disclosure. In general, arrangements are described in sufficient detail to enable those skilled in the art to practice one or more of the aspects, and it should be appreciated that other arrangements may be utilized and that structural, logical, software, electrical and other changes may be made without departing from the scope of the particular aspects. Particular features of one or more of the aspects

described herein may be described with reference to one or more particular aspects or figures that form a part of the present disclosure, and in which are shown, by way of illustration, specific arrangements of one or more of the aspects. It should be appreciated, however, that such features are not limited to usage in the one or more particular aspects or figures with reference to which they are described. The present disclosure is neither a literal description of all arrangements of one or more of the aspects nor a listing of features of one or more of the aspects that must be present in all arrangements.

[0049] Headings of sections provided in this patent application and the title of this patent application are for convenience only, and are not to be taken as limiting the disclosure in any way.

[0050] Devices that are in communication with each other need not be in continuous communication with each other, unless expressly specified otherwise. In addition, devices that are in communication with each other may communicate directly or indirectly through one or more communication means or intermediaries, logical or physical.

[0051] A description of an aspect with several components in communication with each other does not imply that all such components are required. To the contrary, a variety of optional components may be described to illustrate a wide variety of possible aspects and in order to more fully illustrate one or more aspects. Similarly, although process steps, method steps, algorithms or the like may be described in a sequential order, such processes, methods and algorithms may generally be configured to work in alternate orders, unless specifically stated to the contrary. In other words, any sequence or order of steps that may be described in this patent application does not, in and of itself, indicate a requirement that the steps be performed in that order. The steps of described processes may be performed in any order practical. Further, some steps may be performed simultaneously despite being described or implied as occurring non-simultaneously (e.g., because one step is described after the other step). Moreover, the illustration of a process by its depiction in a drawing does not imply that the illustrated process is exclusive of other variations and modifications thereto, does not imply that the illustrated process or any of its steps are necessary to one or more of the aspects, and does not imply that the illustrated process is preferred. Also, steps are generally described once per aspect, but this does not mean they must occur once, or that they may only occur once each time a process, method, or algorithm is carried out or executed. Some steps may be omitted in some aspects or some occurrences, or some steps may be executed more than once in a given aspect or occurrence.

[0052] When a single device or article is described herein, it will be readily apparent that more than one device or article may be used in place of a single device or article. Similarly, where more than one device or article is described herein, it will be readily apparent that a single device or article may be used in place of the more than one device or article.

[0053] The functionality or the features of a device may be alternatively embodied by one or more other devices that are not explicitly described as having such functionality or features. Thus, other aspects need not include the device itself.

[0054] Techniques and mechanisms described or referenced herein will sometimes be described in singular form for clarity. However, it should be appreciated that particular aspects may include multiple iterations of a technique or multiple instantiations of a mechanism unless noted otherwise. Process descriptions or blocks in figures should be understood as representing modules, segments, or portions of code which include one or more executable instructions for implementing specific logical functions or steps in the process. Alternate implementations are included within the scope of various aspects in which, for example, functions may be executed out of order from that shown or discussed, including substantially concurrently or in reverse order, depending on the functionality involved, as would be understood by those having ordinary skill in the art.

Definitions

[0055] As used herein, “graph” is a representation of information and relationships, where each primary unit of information makes up a “node” or “vertex” of the graph and the relationship

between two nodes makes up an edge of the graph. Nodes can be further qualified by the connection of one or more descriptors or “properties” to that node. For example, given the node “James R,” name information for a person, qualifying properties might be “183 cm tall,” “DOB Aug. 13, 1965” and “speaks English”. Similar to the use of properties to further describe the information in a node, a relationship between two nodes that forms an edge can be qualified using a “label”. Thus, given a second node “Thomas G,” an edge between “James R” and “Thomas G” that indicates that the two people know each other might be labeled “knows.” When graph theory notation (Graph=(Vertices, Edges)) is applied this situation, the set of nodes are used as one parameter of the ordered pair, V and the set of 2 element edge endpoints are used as the second parameter of the ordered pair, E. When the order of the edge endpoints within the pairs of E is not significant, for example, the edge James R, Thomas G is equivalent to Thomas G, James R, the graph is designated as “undirected.” Under circumstances when a relationship flows from one node to another in one direction, for example James R is “taller” than Thomas G, the order of the endpoints is significant. Graphs with such edges are designated as “directed.” In the distributed computational graph system, transformations within transformation pipeline are represented as directed graph with each transformation comprising a node and the output messages between transformations comprising edges. Distributed computational graph stipulates the potential use of non-linear transformation pipelines which are programmatically linearized. Such linearization can result in exponential growth of resource consumption. The most sensible approach to overcome possibility is to introduce new transformation pipelines just as they are needed, creating only those that are ready to compute. Such method results in transformation graphs which are highly variable in size and node, edge composition as the system processes data streams. Those familiar with the art will realize that transformation graph may assume many shapes and sizes with a vast topography of edge relationships and node types. It is also important to note that the resource topologies available at a given execution time for a given pipeline may be highly dynamic due to changes in available node or edge types or topologies (e.g. different servers, data centers, devices, network links, etc.) being available, and this is even more so when legal, regulatory, privacy and security considerations are included in a DCG pipeline specification or recipe in the DSL. Since the system can have a range of parameters (e.g. authorized to do transformation x at compute locations of a, b, or c) the JIT, JIC, JIP elements can leverage system state information (about both the processing system and the observed system of interest) and planning or modeling modules to compute at least one parameter set (e.g. execution of pipeline may say based on current conditions use compute location b) at execution time. This may also be done at the highest level or delegated to lower level resources when considering the spectrum from centralized cloud clusters (i.e. higher) to extreme edge (e.g. a wearable, or phone or laptop). The examples given were chosen for illustrative purposes only and represent a small number of the simplest of possibilities. These examples should not be taken to define the possible graphs expected as part of operation of the invention

[0056] As used herein, “transformation” is a function performed on zero or more streams of input data which results in a single stream of output which may or may not then be used as input for another transformation. Transformations may comprise any combination of machine, human or machine-human interactions Transformations need not change data that enters them, one example of this type of transformation would be a storage transformation which would receive input and then act as a queue for that data for subsequent transformations. As implied above, a specific transformation may generate output data in the absence of input data. A time stamp serves as an example. In the invention, transformations are placed into pipelines such that the output of one transformation may serve as an input for another. These pipelines can consist of two or more transformations with the number of transformations limited only by the resources of the system. Historically, transformation pipelines have been linear with each transformation in the pipeline receiving input from one antecedent and providing output to one subsequent with no branching or

iteration. Other pipeline configurations are possible. The invention is designed to permit several of these configurations including, but not limited to: linear, afferent branch, efferent branch and cyclical.

[0057] A “pipeline,” as used herein and interchangeably referred to as a “data pipeline” or a “processing pipeline,” refers to a set of data streaming activities and batch activities. Streaming and batch activities can be connected indiscriminately within a pipeline and compute, transport or storage (including temporary in-memory persistence such as Kafka topics) may be optionally inferred/suggested by the system or may be expressly defined in the pipeline domain specific language. Events will flow through the streaming activity actors in a reactive way. At the junction of a streaming activity to batch activity, there will exist a StreamBatchProtocol data object. This object is responsible for determining when and if the batch process is run. One or more of three possibilities can be used for processing triggers: regular timing interval, every N events, a certain data size or chunk, or optionally an internal (e.g. APM or trace or resource based trigger) or external trigger (e.g. from another user, pipeline, or exogenous service). The events are held in a queue (e.g. Kafka) or similar until processing. Each batch activity may contain a “source” data context (this may be a streaming context if the upstream activities are streaming), and a “destination” data context (which is passed to the next activity). Streaming activities may sometimes have an optional “destination” streaming data context (optional meaning: caching/persistence of events vs. ephemeral). System also contains a database containing all data pipelines as templates, recipes, or as run at execution time to enable post-hoc reconstruction or re-evaluation with a modified topology of the resources (e.g. compute, transport or storage), transformations, or data involved.

Conceptual Architecture

[0058] FIG. **1** is a block diagram illustrating an exemplary system architecture for a distributed generative artificial intelligence reasoning and action platform **120**, according to an embodiment. According to the embodiment, platform **120** is configured as a cloud-based computing platform comprising various system or sub-system components configured to provide functionality directed to the execution of neuro-symbolic generative AI reasoning and action. Exemplary platform systems can include a distributed computational graph (DCG) computing system **121**, a curation computing system **122**, a marketplace computing system **123**, and a context computing system **124**. In some embodiments, systems **121-124** may each be implemented as standalone software applications or as a services/microservices architecture which can be deployed (via platform **120**) to perform a specific task or functionality. In such an arrangement, services can communicate with each other over an appropriate network using lightweight protocols such as HTTP, gRPC, or message queues. This allows for asynchronous and decoupled communication between services. Services may be scaled independently based on demand, which allows for better resource utilization and improved performance. Services may be deployed using containerization technologies such as Docker and orchestrated using container orchestration platforms like Kubernetes. This allows for easier deployment and management of services.

[0059] The distributed generative AI reasoning and action platform **120** can enable a more flexible approach to incorporating machine learning (ML) models into the future of the Internet and software applications; all facilitated by a DCG architecture capable of dynamically selecting, creating, and incorporating trained models with external data sources and marketplaces for data and algorithms.

[0060] According to the embodiment, DCG computing system **121** provides orchestration of complex, user-defined workflows built upon a declarative framework which can allow an enterprise user **110** to construct such workflows using modular components which can be arranged to suit the use case of the enterprise user. As a simple example, an enterprise user **110** can create a workflow such that platform **120** can extract, transform, and load enterprise-specific data to be used as contextual data for creating and training a ML or AI model. The DCG functionality can be

extended such that an enterprise user can create a complex workflow directed to the creation, deployment, and ongoing refinement of a trained model (e.g., LLM). For example, in some embodiments, an enterprise user **110** can select an algorithm from which to create the trained model, and what type of data and from what source they wish to use as training data. DCG computing system **121** can take this information and automatically create the workflow, with all the requisite data pipelines, to enable the retrieval of the appropriate data from the appropriate data sources, the processing/preprocessing of the obtained data to be used as inputs into the selected algorithm(s), the training loop to iteratively train the selected algorithms including model validation and testing steps, deploying the trained model, and finally continuously refining the model over time to improve performance.

[0061] A context computing system **124** is present and configured to receive, retrieve, or otherwise obtain a plurality of context data from various sources including, but not limited to, enterprise users **110**, marketplaces **130***a-n*, third-party sources **150**, and other data sources **140***a-n*. Context computing system **124** may be configured to store obtained contextual data in a data store. For example, context data obtained from various enterprise endpoints **110***a-n* of a first enterprise may be stored separately from the context data obtained from the endpoints of a second enterprise. In some embodiments, context data may be aggregated from multiple enterprises within the same industry and stored as a single corpus of contextual data. In such embodiments, contextual data may be transformed prior to processing and storage so as to protect any potential private information or enterprise-specific secret knowledge that the enterprise does not wish to share.

[0062] A curation computing system **122** is present and configured to provide curated (or not) responses from a trained model (e.g., LLM) to received user queries. A curated response may indicate that it has been filtered, such as to remove personal identifying information or to remove extraneous information from the response, or it may indicate that the response has been augmented with additional context or information relevant to the user. In some embodiments, multiple trained models (e.g., LLMs) may each produce a response to a given prompt, which may include additional contextual data/elements, and a curation step may include selecting a single response of the multiple responses to send to a user, or the curation may involve curating the multiple responses into a single response. The curation of a response may be based on rules or policies that can set an individual user level, an enterprise level, or at a department level for enterprises with multiple departments (e.g., sales, marketing, research, product development, etc.).

[0063] According to the embodiment, an enterprise user **110** may refer to a business organization or company. An enterprise may wish to incorporate a trained ML model into their business processes. An enterprise may comprise a plurality of enterprise endpoints **110***a-n* which can include, but are not limited to, mobile devices, workstations, laptops, personal computers, servers, switches, routers, industrial equipment, gateways, smart wearables, Internet-of-Things (IoT) devices, sensors, and/or the like. An enterprise may engage with platform **120** to create a trained model to integrate with its business processes via one or more enterprise endpoints. To facilitate the creation of purpose-built, trained model, enterprise user **110** can provide a plurality of enterprise knowledge **111** which can be leveraged to build enterprise specific (or even specific to certain departments within the enterprise) ML/AI models. Enterprise knowledge **111** may refer to documents or other information important for the operation and success of an enterprise. Data from internal systems and databases, such as customer relationship management (CRM) systems, enterprise resource planning (ERP) systems, rules and policies databases, and transactional databases, can provide information about the operational context of an enterprise. For example, product knowledge, market knowledge, industry trends, regulatory knowledge, business processes, customer knowledge, technology knowledge, financial knowledge, organization knowledge, and risk management knowledge may be included in enterprise knowledge base **111**.

[0064] According to the embodiment, platform **120** is configured to retrieve, receive, or otherwise obtain a plurality of data from various sources. A plurality of marketplaces **130***a-n* may be present

and configured to provide centralized repositories for data, algorithms, and expert judgment, which can be purchased, sold, or traded on an open marketplace. External data sourced from various marketplaces **130***a-n* can be used as a training data source for creating trained models for a particular use case. A marketplace computing system **123** is present and configured to develop and integrate various marketplaces **130***a-n*. Marketplace computing system **123** can provide functionality directed to the registration of experts or entities. An expert may be someone who has a deep understanding and knowledge of a specific industry, including its trends, challenges, technologies, regulations, and best practices. Industry experts often have many years of experience working in the industry and have developed a reputation for their expertise and insights. Examples of experts can include, but are not limited to, consultants, analysts, researchers, academics, or professionals working in the industry. In some embodiments, experts and/or entities can register with platform **120** so that they may become verified experts/entities. In such an embodiment, an expert/entity profile may be created which can provide information about expert judgment, scored data and algorithms, and comparisons/statistics about the expert's/entity's scores and judgment with respect to other expert/entities. Marketplace computing system **123** may further provide functionality directed to the management of the various marketplaces and the data/algorithms provided therein.

[0065] According to some embodiments, platform **120** can communicate with and obtain data from various third-party services **150**. For example, third-party services can include LLM services such as APIs and LLM hosting platforms, which platform **120** can interface with to obtain algorithms or models to use as starting points for training a neuro-symbolic generative AI reasoning and action model to be deployed at the enterprise or individual level. As another example, social media platforms can provide data about trends, events, and public sentiment, which can be useful for understanding the social context of a situation. Exemplary data sources **140***a-n* can include, but are not limited to, sensors, web data, environmental data, and survey and interviews.

[0066] FIG. **2** is a block diagram illustrating an exemplary aspect of a distributed generative AI reasoning and action platform incorporating various additional contextual data. According to the aspect, a plurality of contextual data from various data sources may be integrated into platform **120**. A simple exemplary directed computational graph **200** is illustrated within the cloud and utilizing the plurality of contextual data to create and train a model. Various marketplaces **130***a-n* are shown which can provide contextual data to platform **120** including an expert judgment marketplace **260** and a model and retrieval augmented generation (RAG) marketplace **220**. According to the aspect, DCG **200** orchestrates model (and model weight) selection **204**, including multi-model usage in series or parallel (i.e., feed output of one model into another, or compare and choose outputs across multiple models), based on multiple data sources (both trained and external), input from crowdsourced expert judgment, training or tuning data set corpora, and RAG libraries.

[0067] Expert judgment will become increasingly important in the world of proprietary or otherwise blackbox ML or AI models where hallucinations and training data quality may produce misleading or otherwise incorrect results. The expert judgment marketplace **260** provides a way for experts **230** to weigh-in on the correctness of data whether that is training data or model output, and can be facilitated by a browser extension **240**, for example, to score things like data sources during their daily “trip around web”. This trip report scoring **250** concept allows experts to score data sources. In an implementation, a browser extension **240** is developed with an accuracy score input where the user can rank a news article they are reading as they consume it. Expert judgment marketplace **260** allows for consumers to pick and rank “experts” based on how well their judgment helps or hinders their overall consumption of model output. For example, experts that routinely highly rank data sources, like news sites, that are known to spread false information should likewise be less trusted over time compared to their peers, and any models trained on that data similarly less trusted. Ultimately a database **270** of data sources and schemas scored by algorithms or experts could be used as input into the DCG **200** for more accurate and real-time

inference based on ongoing rating of preferred data set and data format combinations (e.g. the same data might be purchased in unstructured, structured, schematized, normalized, or semantified formats) which may introduce different types of bias or impacts on performance, results, or processing costs.

[0068] Accordingly, a RAG marketplace **220** may be implemented to further refine model output. RAG information may be included as additional context which can be supplied to a GenAI model in addition to a prompt (engineered, or otherwise). This is especially important where companies may want to sell access to their proprietary dataset through the form of a RAG. For example, a medical research company may have valuable information they could sell to other institutions in the form of a RAG to augment related research without specifically providing access to the raw training data. Retrieval-augmented generation is a framework that combines elements of retrieval-based and generative models to improve the performance of natural language processing tasks. In RAG, a retriever component is used to select relevant information from a large corpus, and a generator component is used to produce a final output based on both the retrieved information and the input query. RAG marketplace **220** may be scored by experts for accuracy and effectiveness across domains.

[0069] According to the aspect, a user experience curation engine **210** is needed that is able to curate output whether that is in the form of filtering out sensitive data or simply customizing results in a way the user prefers (which may be based on user-/entity-defined rules or policies). A user can submit a query to experience curation engine **210** which can send the query to the DCG trained model to obtain a response. Experience curation **210** may then process the received response to curate it (or not) to meet the preferences of the user.

[0070] As illustrated, DCG **200** shows a simple example of a directed computational graph which can be used to create a complex workflow to create and train an MI/AI model (e.g., variations of or standard transformer architecture). A shown, the DCG comprises multiple sources of information for training the selected models(s) including multiple data sources **201***a-n* which may or may not be scored by experts, expert judgment **202**, and one or more RAGs **203** which may be obtained from RAG marketplace **220** or may be obtained directly from enterprise knowledge. DCG may have access to stored models or variants thereof. In the illustration, LLAMA (Learned Layer-wise Attention Metric for Transformers), PALM (Permuted Adaptive Lateral Modulation), and HYENA (Hyperbolic Encoder for Efficient Attention) are shown as possible examples of the types of models which can be selected by the DCG to create and train a GenAI model. Furthermore, the "model parameters" and mathematical techniques or assumptions used in each model may be cataloged and included in a model-specific template which may be stored in cloud-based storage on platform **120**. In some embodiments, platform **120** may store a hierarchical representation of transformer models (e.g., as a graph), which may represent a lineage of the evolution of transformer models. In an implementation, model selection or exploration involves selections based on the evolutionary tree of one or more model types and use said tree (e.g., graph) for selections in heuristic search for best algorithm/data combinations, licensing costs/explorations, etc. It should be appreciated that certain aspects of the invention may be tailored based on what kind of mathematical approach underpins a specific model.

[0071] In operation, DCG **200** obtains the various contextual data from the connected data sources, creates training, validation, and test datasets from the obtained data, and uses the various datasets to train, validate, and test the model as it undergoes a model training loop that iteratively trains the model to generate responses based on the plurality of contextual data.

[0072] FIG. **3** is a diagram illustrating incorporating symbolic reasoning in support of LLM-based generative AI, according to an aspect of a neuro-symbolic generative AI reasoning and action platform. According to the aspect, platform **120** can incorporate symbolic reasoning and in-context learning to create and train off the shelf models (e.g., an LLM foundational model or narrow model) through clever prompting and conditioning on private data or very situation specific

"contextual" data. Platform **120** can obtain contextual data **301** and preprocess the data for storage. Contextual data **301** may refer to data obtained from marketplaces **130***a-n*, third-party services **150**, and enterprise knowledge **111**, as well as other types of contextual data that may be obtained from other sources. DCG **330** is responsible for orchestrating the entire process and can create data pipelines **310** as needed to facilitate the ingestion of contextual data **301**. Contextual data can include text documents, PDFs, and even structure formats like CSV (comma-separated values) or SQL tables or other common generic data formats like OWL or RDF or domain specific content such as the Financial Industry Business Ontology (FIBO) or Open Graph of Information Technology (OGIT). This stage involves storing private data (e.g., context data) to be retrieved later.

[0073] Typically, the context data **301** is broken into chunks, passed through and embedding model **315**, then stored in a specialized database called a vector database **320**. Embedding models are a class of models used in many tasks such as natural language processing (NLP) to convert words, phrases, or documents into numerical representations (embeddings) that capture similarity which often correlates semantic meaning. Exemplary embedding models can include, but are not limited to, text-embedding-ada-002 model (i.e., OpenAI API), bidirectional encoder representations form transformers, Word2Vec, FastText, transformer-based models, and/or the like. The vector database **315** is responsible for efficiently storing, comparing, and retrieving a large plurality of embeddings (i.e., vectors). Vector database **315** may be any suitable vector database system known to those with skill in the art including, but not limited to, open source systems like Pinecone, Weaviate, Vespa, and Qdrant. According to the embodiment, embedding model **315** may also receive a user query from experience curation **340** and vectorize it where it may be stored in vector database **320**. This provides another useful datapoint to provide deeper context when comparing received queries against stored query embeddings.

[0074] A user may submit a query **303** to an experience curation engine **340** which starts the prompt construction and retrieval process. The query is sent to DCG **330** which can send the query to various components such as prompt engineering **325** and embedding model **315**. Embedding model **315** receives the query and vectorizes it and stores it in vector database **320**. The vector database **320** can send contextual data (via vectors) to DCG **330** and to various APIs/plugins **335**. Prompt engineering **325** can receive prompts **302** from developers to train the model on. These can include some sample outputs such as in few-shot prompting. The addition of prompts via prompt engineering **325** is designed to ground model responses in some source of truth and provide external context the model wasn't trained on. Other examples of prompt engineering that may be implemented in various embodiments include, but are not limited to, chain-of-thought, self-consistency, generated knowledge, tree of thoughts, directional stimulus, and/or the like.

[0075] During a prompt execution process, experience curation **340** can send user query to DCG **330** which can orchestrate the retrieval of context and a response. Using its declarative roots, DCG **330** can abstract away many of the details of prompt chaining; interfacing with external APIs **335** (including determining when an API call is needed); retrieving contextual data from vector databases **330**; and maintaining memory across multiple LLM calls. The DCG output may be a prompt, or series of prompts, to submit to a language model via LLM services **360** (which may be potentially prompt tuned). In turn, the LLM processes the prompts, contextual data, and user query to generate a contextually aware response which can be sent to experience curation **340** where the response may be curated, or not, and returned to the user as output **304**.

[0076] FIG. **4** is a block diagram illustrating an exemplary architecture for a neuro-symbolic generative AI reasoning and action platform **400** configured for federated learning at a plurality of edge devices **410***a-n*, according to an embodiment. According to the embodiment, platform **400** comprises DCH computing system **421**, curation computing system **422**, marketplace computing system **423**, and context computing system **424**. According to an embodiment, edge devices **410***a-n* may represent various enterprise endpoints. In other embodiments, edge devices **410***a-n* may

represent various endpoints from two or more separate enterprises. In an embodiment, an edge device **410***a-n* may be a computing device associated with a platform user, such as someone who engages with the platform for experience curation or an expert who provides expert judgment scores to platform **400** via, for example, expert judgment marketplace **260** or some other mechanism.

[0077] As shown, each edge device **410***a-n* may comprise instances of local models **411***a-n*, context classification processes **412**-*n*, and experience curation processes **413***a* operating on the device. Each edge device may have access to a local data or knowledge base **420***a-n* and which is only accessible by its associated edge device. Edge devices **410***a-n* may utilize these components to perform various computations wherein the processing of data and execution of algorithms happens locally on the device, rather than relying on the systems and services provided by platform **400**. In some embodiments, a plurality of edge devices **410***a-n* may be implemented as individual computing nodes in a decentralized federated system, wherein tasks and data may be distributed across multiple nodes, allowing for parallel processing and potentially faster computation. Federated systems are often used in scenarios where data privacy and security are important, as data can remain on local nodes and only aggregated or processed results are shared more widely.

[0078] In some implementations, the platform **400** may leverage federated learning, where machine learning models **411***a-n* are trained across multiple decentralized edge devices **410***a-n*, with the models' updates being aggregated centrally. This approach allows for the training of models without the need to centrally store sensitive data from individual devices. For example, each edge device **410***a-n* could train local instances of neuro-symbolic GenAI reasoning and action models and local instances of context classification models **412***a-n*. According to an embodiment, context classification models **412***a-n* may be configured to select relevant passages from a knowledge base **420***a-n* or corpus given a query. This can be done using various techniques such as BM25, TF-IDF, or neural retrieval models like dense passage retrieval. The retrieved passages serve as context or input to a generator (e.g., a transformer-based model).

[0079] Federated learning can occur at the edge device wherein the context classification model **412***a* is trained locally. Periodically, (e.g., hourly, daily, weekly, etc.) platform **400** may collect (e.g., aggregate) model parameters, encrypted data, and/or the like from all of, or a subset of, edge devices **410***a-n* and apply the aggregated model parameters as an update to a master or global model (e.g., context classification, neuro-symbolic GenAI model, etc.). The updated global model or just its parameters, may be transmitted to all of, or a subset of, the edge devices **410***a-n* where they may be applied to the local models operating thereon. Similarly, platform **400** can aggregate obtained training data, which may or may not be encrypted, and apply the training data to global models. These updated models may be transmitted to edge devices as described above.

[0080] As shown, edge devices **410***a-n* may further comprise a curation application **413***a-n* operating on the device. Curation application **413***a* may be configured to act as an intermediary between a user who can submit a query and models **411***a* which receive the query and generate a response back. Curation **413***a-n* may receive a response from a locally stored model and curate the response based on user (or entity) defined rules or preferences. For example, a response may first be filtered of any personal information by curation **413***a* prior to the being relayed back to the user. As another example, curation **413***a* may transform the response into specific format, style, or language based on user defined preferences. This allows the edge device **410***a* user to have their experience with the local models curated to fit any criteria they deem important.

[0081] FIG. **5** is a block diagram illustrating an exemplary architecture for a neuro-symbolic generative AI reasoning and action platform **500** configured to utilize a midserver **530** to act as a computing intermediary between a plurality of edge devices **510***a-n* and the platform. According to the embodiment, midserver **530** facilitates communication between edge devices **510***a-n* and the backend systems **521**, **522**, **523**, **524** provided by platform **500**. According to the embodiment, midserver **530** may have stored and operating on it one or more neuro-symbolic GenAI reasoning

and action models **531**, context classification processes **532**, and curation processes **533**. Midserver **530** can be configured to periodically receive data (e.g., context data) and state information from each of the connected edge devices **510***a-n*. Midserver **530** may use this information to train/update the models **531**, **532**. Additionally, midserver **530** can be configured to receive user-submitted queries from edge devices via curation **533**, obtain relevant context associated with the received query via context classification **532**, and use a neuro-symbolic GenAI model **531** to process the query and context data to generate a response to the user. The generated response may be curated (or not) and transmitted back to the user of the edge device.

[0082] In some implementations, edge devices **510***a-n* may have stored upon them local models as described in FIG. **4**, and midserver **530** may store global, models or even mid-tier models associated with the local models. In such an implementation, midserver can aggregate model parameters and update the global/mid-tier models accordingly.

[0083] FIG. **6** is a block diagram illustrating an exemplary mobile device **610***a-n* configured for experience curation using embedded capabilities and functionality provided by a neuro-symbolic generative AI reasoning and action platform **600**, according to an embodiment. According to the embodiment, a mobile device **610***a* may comprise an operating system **611**, various software applications **612** (e.g., text messaging application, social media application, mobile games, music streaming applications, etc.), a local instance of a neuro-symbolic GenAI model **613**, a context classification model **614**, and an experience curation application **615**. Mobile devices **610***a-n* may further comprise a processor, memory, sensors, storage, wireless communication modules, a display, audio components, and various other components to enable the functionality of a mobile computing device. Mobile devices **610***a-n* may connect to platform **600** via a suitable communication network such as the Internet. In some embodiments, mobile device may utilize the systems and services **621**, **622**, **623**, **624** provided by platform to facilitate query-response interactions with a neuro-symbolic GenAI model.

[0084] According to the embodiment, mobile device **610***a* stores and operates local models **613**, **614** and a curation application **615** which can be leveraged during instances when mobile device **610***a* is unable to connect with platform **600** or otherwise has an intermittent connection thereby making data transmission difficult, slow, or impossible. In such situations, mobile device **610***a* can leverage the local components to perform computation at the edge. A user of mobile device **610***a* can use curation application **615** to submit a query to the local neuro-symbolic GenAI model **613**, along with any aggregated context retrieved via context classification **614**. The model **613** can generate a response and send it to curation application **615** where it may be curated (or not) based on the mobile device user's preferences or rules.

[0085] In some embodiments, when there is only an intermittent connection to platform **600**, such as when a mobile device is in an area with poor network coverage, various strategies may be implemented to provide functionality to the mobile device user. For example, data (e.g., a user submitted query or prompt) can be temporarily stored in a buffer on the device until a connection to platform **600** is available. Once the connection is reestablished, the buffered data can be transmitted. Likewise, frequently accessed data or recently transmitted data can be cached on the device. This allows the device to access the data locally when a connection to platform **600** is not available. In some implementations, data can be compressed before transmission to reduce the amount of data that needs to be transmitted. This can help to minimize the impact of intermittent connections on data transmission. In some embodiments, mobile device **610***a-n* may use protocols that are designed to handle intermittent connections, such as MQTT (Message Queuing Telemetry Transport) or CoAP (Constrained Application Protocol), can help to ensure that data is successfully transmitted even in challenging network conditions. Finally, some use cases may implement an offline mode that allows users to continue using the application (or local instances) and storing data locally until a connection to platform **600** is available again.

[0086] FIG. **7** is a block diagram illustrating an exemplary aspect of a distributed generative

artificial intelligence reasoning and action platform, a curation computing system **700**. According to the aspect, curation computing system **700** is configured to provide curated (or not) responses from a trained model (e.g., transformer-based model) to received user queries. A curated response may indicate that the response has been filtered, such as to remove personal identifying information or to remove extraneous information from the response, or it may indicate that the response has been augmented with additional context or information relevant to the user. The curation of a response may be based on rules or policies that can be set at an individual user level, an enterprise level, or at a department level for enterprises with multiple departments (e.g., sales, marketing, research, product development, etc.). User/entity rules and/or preferences may be stored in a data storage system of platform **120** and retrieved by a rules management component **740** during experience curation processes.

[0087] In operation, curation computing **700** receives a user query **701** directed to a neuro-symbolic GenAI model. A query portal **710** may be present and configured to receive a query **701** and prepare it for processing by a GenAI model. For example, a query may be split into tokens, (e.g., words or sub words) which are basic units of the language model. As another example, a text-based query may undergo normalization (e.g., converting to lowercase, removing punctuation, handling special characters, etc.) to ensure consistency and improve model performance. As yet another example, for models that use attention mechanisms, an attention mask may be applied to the input to indicate which tokens should be attended to and which should be ignored. In some implementations, a query portal **710** may be configured to send received queries to an embedding model which can vectorize the received query and store it in a vector database. In such embodiments, stored query embeddings may be used as a form of contextual data which may be retrieved and transmitted with the query to a GenAI model which generates a response based on the received query and contextual data.

[0088] According to the aspect, a response portal **720** is present and configured to receive a response from one a GenAI model and a response management system **730** determines if the received response needs to be curated or not. If the response does not need to be curated, then it may be sent as an uncrated response **702** to the user who submitted the query. Response management **730** can determine if there are any user/entity defined rules or preferences available such as stored in a user/entity profile in a data storage system of platform **120**. Rules management **740** can retrieve said rules and response management can curate or otherwise augment the received response based on the user/entity rules or preferences. The result is a curated response **702** which can be transmitted back to the user who submitted the query.

[0089] FIG. **8** is a block diagram illustrating an exemplary aspect of a distributed generative artificial intelligence reasoning and action platform, a marketplace computing system **800**. According to the aspect, marketplace computing system **800** is present and configured to develop and integrate various marketplaces **130***a-n* for data, algorithms, and RAGs into platform **120**. Marketplace computing system **800** can provide functionality directed to the registration of experts **810** or entities. An expert may be someone who has a deep understanding and knowledge of a specific industry, including its trends, challenges, technologies, regulations, and best practices. Industry experts often have many years of experience working in the industry and have developed a reputation for their expertise and insights. An expert may be registered by providing proof of identity and qualifications, and creating an expert profile which can store a variety of information about the expert such as their name, industry, credentials, scores (e.g., scores that the expert has assigned to data sources, models/algorithms, model outputs, and/or the like), and reputation. For example, a university professor who specializes in transformer-based algorithms can register as an expert in the realm of generative algorithms. As another example, a virologist could register as an expert and provide scores for academic papers which disclose a new methodology for viral spread modelling.

[0090] Marketplace computing system **800** may further comprise a market management component

820 which can interface with a plurality of markets **130***a-n* to integrate information contained therein. A scored data management component **830** may be configured to interface with a browser extension **240** or expert judgment marketplace **260** to retrieve expert scores and store them in an expert judgment score database **270**. According to the aspect, an algorithm management component **840** is present and configured to acquire algorithms from algorithm marketplaces to be used in the construction and configuration of neuro-symbolic GenAI models.

[0091] FIG. **9** is a block diagram illustrating a simple example of a distributed computational graph **900** representation for providing neuro-symbolic GenAI capabilities, according to an aspect. According to the aspect, the DCG may be represented as a series of nodes which represent discrete computational or data processing functions, and a series of edges connecting the nodes which represent information or data messages being sent between processing nodes. A DCG can be used to acquire a plurality of context data in the form of an enterprise knowledge base **910**. A data transformation node **920** is created to handle the ingestion and transformation of acquired context data. Obtained data may then be sent to a data embedding node **930** which can vectorize the received context data. The vectorized data may flow from the embedding node **930** to a data storage node **950**. Data storage node **950** may select the appropriate vector database **980** in which to store the vectorized context data. An input node **940** may allow for a user to submit a query to the workflow. The user query can be sent to data embedding node **930** where it may be vectorized and sent to data storage node **950** for storage in the vector database. The user query can also be sent to a model node **960** which contains the selected model(s) which will process the user query along with any relevant context data obtained from data storage node vector database **980**. Model node **960** then processes this information to generate a response which can be sent to output node **970**. In some instances, output node **970** may output the response directly to the user. In other instances, output node **970** may be configured to transform the response into a curated response based on user/entity defined rules or preferences.

[0092] FIGS. **10-14** illustrate various exemplary aspects of system architectures of distributed computational graph computing environments. For more detailed information regarding the operation of the various components and aspects described herein with respect to FIGS. **10-14**, please refer to U.S. patent application Ser. No. 15/931,534 which is incorporated herein by reference.

[0093] FIG. **10** is a block diagram illustrating an exemplary aspect of an embodiment of a distributed computational graph computing system utilizing an advanced cyber decision platform (ACDP) for external network reconnaissance and contextual data collection. Client access to the system **1005** for specific data entry, system control and for interaction with system output such as automated predictive decision making and planning and alternate pathway simulations, occurs through the system's distributed, extensible high bandwidth cloud interface **1010** which uses a versatile, robust web application driven interface for both input and display of client-facing information via network **1007** and operates a data store **1012** such as, but not limited to MONGODB™, COUCHDB™, CASSANDRA™ or REDIS™ according to various arrangements. Much of the enterprise knowledge/context data analyzed by the system both from sources within the confines of the enterprise business, and from cloud based sources, also enter the system through the cloud interface **1010**, data being passed to the connector module **1035** which may possess the API routines **1035***a* needed to accept and convert the external data and then pass the normalized information to other analysis and transformation components of the system, the directed computational graph module **1055**, high volume web crawler module **1015**, multidimensional time series database (MDTSDB) **1020** and the graph stack service **1045**. The directed computational graph module **1055** retrieves one or more streams of data from a plurality of sources, which includes, but is in no way not limited to, enterprise knowledge, RAGs, expert judgment/scores, a plurality of physical sensors, network service providers, web based questionnaires and surveys, monitoring of electronic infrastructure, crowdsourcing campaigns, and human input device

information. Within the directed computational graph module **1055**, data may be split into two identical streams in a specialized pre-programmed data pipeline **1055***a*, wherein one sub-stream may be sent for batch processing and storage while the other sub-stream may be reformatted for transformation pipeline analysis. The data is then transferred to the general transformer service module **1060** for linear data transformation as part of analysis or the decomposable transformer service module **1050** for branching or iterative transformations that are part of analysis. The directed computational graph module **1055** can represent all data as directed graphs where the transformations are nodes and the result messages between transformations edges of the graph. The high volume web crawling module **1015** uses multiple server hosted preprogrammed web spiders, which while autonomously configured are deployed within a web scraping framework **1015***a* of which SCRAPY™ is an example, to identify and retrieve data of interest from web based sources that are not well tagged by conventional web crawling technology. Data persistence stores such as the multiple dimension time series data store module **1020** may receive streaming data from a large plurality of sensors that may be of several different types. The multiple dimension time series data store module may also store any time series data encountered by the system such as but not limited to enterprise network usage data, component and system logs, environmental context, edge device state information, performance data, network service information captures such as, but not limited to news and financial feeds, and sales and service related customer data. The module is designed to accommodate irregular and high volume surges by dynamically allocating network bandwidth and server processing channels to process the incoming data. Inclusion of programming wrappers **1020***a* for languages examples of which are, but not limited to C++, PERL, PYTHON, Rust, GoLang, and ERLANG™ allows sophisticated programming logic to be added to the default function of the multidimensional time series database **1020** without intimate knowledge of the core programming, greatly extending breadth of function. Data retrieved by various data stores such as SQL, graph, key-value, or the multidimensional time series database (MDTSDB) **1020** and the high volume web crawling module **1015** may be further analyzed and transformed into task optimized results by the directed computational graph **1055** and associated general transformer service **1050** and decomposable transformer service **1060** modules. Alternately, data from the multidimensional time series database and high volume web crawling modules may be sent, often with scripted cuing information determining important vertexes **1045***a*, to the graph stack service module **1045** which, employing standardized protocols for converting streams of information into graph representations of that data, for example, open graph internet technology although the invention is not reliant on any one standard. Through the steps, the graph stack service module **1045** represents data in graphical form influenced by any predetermined scripted modifications **1045***a* and stores it in a graph-based data store **1045***b* such as GIRAPH™ or a key value pair type data store REDIS™, or RIAK™, among others, all of which are suitable for storing graph-based information.

[0094] Results of the transformative analysis process may then be combined with further client directives, and additional business rules and practices relevant to the analysis and situational information external to the already available data in the automated planning service module **1030** which also runs powerful information theory **1030***a* based predictive statistics functions and machine learning algorithms to allow future trends and outcomes to be rapidly forecast based upon the current system derived results and choosing each a plurality of possible business decisions. Using all available data, the automated planning service module **1030** may propose business decisions most likely to result is the most favorable business outcome with a usably high level of certainty. Closely related to the automated planning service module in the use of system derived results in conjunction with possible externally supplied additional information (i.e., context) in the assistance of end user business decision making, the action outcome simulation module **1025** with its discrete event simulator programming module **1025***a* coupled with the end user facing observation and state estimation service **1040** which is highly scriptable **1040***b* as circumstances

require and has a game engine **1040***a* to more realistically stage possible outcomes of business decisions under consideration, allows business decision makers to investigate the probable outcomes of choosing one pending course of action over another based upon analysis of the current available data.

[0095] FIG. **11** is a block diagram illustrating another exemplary aspect of an embodiment **1100** of a distributed computational graph computing systems utilizing an advanced cyber decision platform. According to the aspect the integrated platform **1100**, is very well suited to perform advanced predictive analytics and predictive simulations to produce investment predictions. Much of the trading specific programming functions are added to the automated planning service module **1030** of the modified advanced cyber decision platform **1100** to specialize it to perform trading analytics. Specialized purpose libraries may include but are not limited to financial markets functions libraries **1151**, Monte-Carlo risk routines **1152**, numeric analysis libraries **1153**, deep learning libraries **1154**, contract manipulation functions **1155**, money handling functions **1156**, Monte-Carlo search libraries **1157**, and quant approach securities routines **1158**. Pre-existing deep learning routines including information theory statistics engine **1159** may also be used. The invention may also make use of other libraries and capabilities that are known to those skilled in the art as instrumental in the regulated trade of items of worth. Data from a plurality of sources used in trade analysis are retrieved, much of it from remote, cloud resident **1101** servers through the system's distributed, extensible high bandwidth cloud interface **110** using the system's connector module **135** which is specifically designed to accept data from a number of information services both public and private through interfaces to those service's applications using its messaging service **135***a* routines, due to case of programming, are augmented with interactive broker functions **1135**, market data source plugins **1136**, e-commerce messaging interpreters **1137**, business-practice aware email reader **1138** and programming libraries to extract information from video data sources **1139**.

[0096] Other modules that make up the advanced cyber decision platform may also perform significant analytical transformations on trade related data. These may include the multidimensional time series data store **1020** with its robust scripting features which may include a distributive friendly, fault-tolerant, real-time, continuous run prioritizing, programming platform such as, but not limited to Erlang/OTP **1121** and a compatible but comprehensive and proven library of math functions of which the C++ math libraries are an example **1122**, data formalization and ability to capture time series data including irregularly transmitted, burst data; the GraphStack service **145** which transforms data into graphical representations for relational analysis and may use packages for graph format data storage such as Titan **1145** or the like and a highly interface accessible programming interface an example of which may be Akka/Spray, although other, similar, combinations may equally serve the same purpose in this role **1146** to facilitate optimal data handling; the directed computational graph module **155** and its distributed data pipeline **155***a* supplying related general transformer service module **160** and decomposable transformer module **150** which may efficiently carry out linear, branched, and recursive transformation pipelines during trading data analysis may be programmed with multiple trade related functions involved in predictive analytics of the received trade data. Both possibly during and following predictive analyses carried out by the system, results must be presented to clients **1005** in formats best suited to convey both important results for analysts to make highly informed decisions and, when needed, interim or final data in summary and potentially raw for direct human analysis. Simulations which may use data from a plurality of field spanning sources to predict future trade conditions these are accomplished within the action outcome simulation module **1025**. Data and simulation formatting may be completed or performed by the observation and state estimation service **1040** using its case of scripting and gaming engine to produce optimal presentation results.

[0097] In cases where there are both large amounts of data to be ingested, schematized, normalized, semantified or otherwise cleansed, enriched or formalized and then intricate transformations such

as those that may be associated with deep machine learning, predictive analytics and predictive simulations, distribution of computer resources to a plurality of systems may be routinely required to accomplish these tasks due to the volume of data being handled and acted upon. The advanced cyber decision platform employs a distributed architecture that is highly extensible to meet these needs. A number of the tasks carried out by the system are extremely processor intensive and for these, the highly integrated process of hardware clustering of systems, possibly of a specific hardware architecture particularly suited to the calculations inherent in the task, is desirable, if not required for timely completion. The system includes a computational clustering module **1180** to allow the configuration and management of such clusters during application of the advanced cyber decision platform. While the computational clustering module is drawn directly connected to specific co-modules of the advanced cyber decision platform these connections, while logical, are for ease of illustration and those skilled in the art will realize that the functions attributed to specific modules of an embodiment may require clustered computing under one use case and not under others. Similarly, the functions designated to a clustered configuration may be role, if not run, dictated. Further, not all use cases or data runs may use clustering.

[0098] FIG. **12** is a diagram of an exemplary architecture for a system for rapid predictive analysis of very large data sets using an actor-driven distributed computational graph **1200**, according to one aspect. According to the aspect, a DCG **1200** may comprise a pipeline orchestrator **1201** that may be used to perform a variety of data transformation functions on data within a processing pipeline, and may be used with a messaging system **1210** that enables communication with any number of various services and protocols, relaying messages and translating them as needed into protocol-specific API system calls for interoperability with external systems (rather than requiring a particular protocol or service to be integrated into a DCG **1200**).

[0099] Pipeline orchestrator **1201** may spawn a plurality of child pipeline clusters **1202***a-b*, which may be used as dedicated workers for streamlining parallel processing. In some arrangements, an entire data processing pipeline may be passed to a child cluster **1202***a* for handling, rather than individual processing tasks, enabling each child cluster **1202***a-b* to handle an entire data pipeline in a dedicated fashion to maintain isolated processing of different pipelines using different cluster nodes **1202***a-b*. Pipeline orchestrator **1201** may provide a software API for starting, stopping, submitting, or saving pipelines. When a pipeline is started, pipeline orchestrator **1201** may send the pipeline information to an available worker node **1202***a-b*, for example using AKKA™ clustering. For each pipeline initialized by pipeline orchestrator **1201**, a reporting object with status information may be maintained. Streaming activities may report the last time an event was processed, and the number of events processed. Batch activities may report status messages as they occur. Pipeline orchestrator **1201** may perform batch caching using, for example, an IGFS™ caching filesystem. This allows activities **1212***a-d* within a pipeline **1202***a-b* to pass data contexts to one another, with any necessary parameter configurations.

[0100] A pipeline manager **1211***a-b* may be spawned for every new running pipeline, and may be used to send activity, status, lifecycle, and event count information to the pipeline orchestrator **1201**. Within a particular pipeline, a plurality of activity actors **1212***a-d* may be created by a pipeline manager **1211***a-b* to handle individual tasks, and provide output to data services **1222***a-d*. Data models used in a given pipeline may be determined by the specific pipeline and activities, as directed by a pipeline manager **1211***a-b*. Each pipeline manager **1211***a-b* controls and directs the operation of any activity actors **1212***a-d* spawned by it. A pipeline process may need to coordinate streaming data between tasks. For this, a pipeline manager **1211***a-b* may spawn service connectors to dynamically create TCP connections between activity instances **1212***a-d*. Data contexts may be maintained for each individual activity **1212***a-d*, and may be cached for provision to other activities **1212***a-d* as needed. A data context defines how an activity accesses information, and an activity **1212***a-d* may process data or simply forward it to a next step. Forwarding data between pipeline steps may route data through a streaming context or batch context.

[0101] A client service cluster **1230** may operate a plurality of service actors **1221***a-d* to serve the requests of activity actors **1212***a-d*, ideally maintaining enough service actors **1221***a-d* to support each activity per the service type. These may also be arranged within service clusters **1220***a-d*, in a manner similar to the logical organization of activity actors **1212***a-d* within clusters **1202***a-b* in a data pipeline. A logging service **1230** may be used to log and sample DCG requests and messages during operation while notification service **1240** may be used to receive alerts and other notifications during operation (for example to alert on errors, which may then be diagnosed by reviewing records from logging service **1230**), and by being connected externally to messaging system **1210**, logging and notification services can be added, removed, or modified during operation without impacting DCG **1200**. A plurality of DCG protocols **1250***a-b* may be used to provide structured messaging between a DCG **1200** and messaging system **1210**, or to enable messaging system **1210** to distribute DCG messages across service clusters **1220***a-d* as shown. A service protocol **1260** may be used to define service interactions so that a DCG **1200** may be modified without impacting service implementations. In this manner it can be appreciated that the overall structure of a system using an actor driven DCG **1200** operates in a modular fashion, enabling modification and substitution of various components without impacting other operations or requiring additional reconfiguration.

[0102] FIG. **13** is a diagram of an exemplary architecture for a system for rapid predictive analysis of very large data sets using an actor-driven distributed computational graph **1200**, according to one aspect. According to the aspect, a variant messaging arrangement may utilize messaging system **1210** as a messaging broker using a streaming protocol **1310**, transmitting and receiving messages immediately using messaging system **1210** as a message broker to bridge communication between service actors **1221***a-b* as needed. Alternately, individual services **1222***a-b* may communicate directly in a batch context **1320**, using a data context service **1330** as a broker to batch-process and relay messages between services **1222***a-b*.

[0103] FIG. **14** is a diagram of an exemplary architecture for a system for rapid predictive analysis of very large data sets using an actor-driven distributed computational graph **1200**, according to one aspect. According to the aspect, a variant messaging arrangement may utilize a service connector **1410** as a central message broker between a plurality of service actors **1221***a-b*, bridging messages in a streaming context **1310** while a data context service **1330** continues to provide direct peer-to-peer messaging between individual services **1222***a-b* in a batch context **1320**.

[0104] It should be appreciated that various combinations and arrangements of the system variants described above (referring to FIGS. **10-14**) may be possible, for example using one particular messaging arrangement for one data pipeline directed by a pipeline manager **1211***a-b*, while another pipeline may utilize a different messaging arrangement (or may not utilize messaging at all). In this manner, a single DCG **1200** and pipeline orchestrator **1201** may operate individual pipelines in the manner that is most suited to their particular needs, with dynamic arrangements being made possible through design modularity as described above in FIG. **12**.

[0105] FIGS. **15-17** illustrate various exemplary aspects of system architectures and methods of distributed computational graph computing environments. For more detailed information regarding the operation of the various components and aspects described herein with respect to FIGS. **15-17**, please refer to U.S. patent application Ser. No. 15/616,427 which is incorporated herein by reference.

[0106] FIG. **15** is a block diagram of an architecture for a transformation pipeline within a system for predictive analysis of very large data sets using distributed computational graph computing system **1500**. According to the aspect, streaming input from a data filter software module, **1505** serves as input to the first transformation node **1510** of the transformation pipeline. Each transformation node's function **1510**, **1520**, **1530**, **1540**, **1550** is performed on input data stream and transformed output message **1515**, **1525**, **1535**, **1545**, **1555**, **1565** is sent to the next step. In this aspect, transformation node **2 1520** has a second input stream **1560**. The specific source of this

input is inconsequential to the operation of the invention and could be another transformation pipeline software module, a data store, human interaction, physical sensors, monitoring equipment for other electronic systems or a stream from the internet as from a crowdsourcing campaign, just to name a few possibilities **1560**. For example, a first input stream may comprise enterprise knowledge and a second input stream may comprise RAG data from a RAG marketplace. Functional integration of a second input stream into one transformation node requires the two input stream events be serialized. The illustrated system can perform this serialization using a decomposable transformation software module. While transformation nodes are described according to various aspects as uniform shape, such uniformity is used for presentation simplicity and clarity and does not reflect necessary operational similarity between transformations within the pipeline. It should be appreciated that one knowledgeable in the field will realize that certain transformations in a pipeline may be entirely self-contained; certain transformations may involve direct human interaction, such as selection via dial or dials, positioning of switch or switches, or parameters set on control display, all of which may change during analysis; other transformations may require external aggregation or correlation services or may rely on remote procedure calls to synchronous or asynchronous analysis engines as might occur in simulations among a plurality of other possibilities. For example, engines may be singletons (composed of a single activity or transformation). Furthermore, leveraging the architecture in this way allows for versioning and functional decomposition (i.e. embedding entire saved workflows as single nodes in other workflows). Further according to the aspect, individual transformation nodes in one pipeline may represent function of another transformation pipeline. It should be appreciated that the node length of transformation pipelines depicted in no way confines the transformation pipelines employed by the invention to an arbitrary maximum length **1510**, **1520**, **1530**, **1540**, **1550**, as, being distributed, the number of transformations would be limited by the resources made available to each implementation of the invention. It should be further appreciated that there need be no limits on transform pipeline length. Output of the last transformation node and by extension, the transform pipeline, **1550** may be sent back to messaging software module **562** for pre-decided action.

Detailed Description of Exemplary Aspects

[0107] FIG. **16** is a process flow diagram of a method **1600** for predictive analysis of very large data sets using the distributed computational graph. One or more streams of data from a plurality of sources, which includes, but is in no way not limited to, a number of physical sensors, web-based questionnaires and surveys, monitoring of electronic infrastructure, crowd sourcing campaigns, and direct human interaction, may be received by system **1601**. The received stream is filtered **1602** to exclude data that has been corrupted, data that is incomplete or misconfigured and therefore unusable, data that may be intact but nonsensical within the context of the analyses being run, as well as a plurality of predetermined analysis related and unrelated criteria set by the authors. Filtered data may be split into two identical streams at this point (second stream not depicted for simplicity), wherein one substream may be sent for batch processing **1600** while another substream may be formalized **1603** for transformation pipeline analysis **1604**, **1500**, and retraining **1605**. Data formalization for transformation pipeline analysis acts to reformat the stream data for optimal, reliable use during analysis. Reformatting might entail, but is not limited to: setting data field order, standardizing measurement units if choices are given, splitting complex information into multiple simpler fields, and stripping unwanted characters, again, just to name a few simple examples. The formalized data stream may be subjected to one or more transformations. Each transformation acts as a function on the data and may or may not change the data. Within the invention, transformations working on the same data stream where the output of one transformation acts as the input to the next are represented as transformation pipelines. While the great majority of transformations in transformation pipelines receive a single stream of input, modify the data within the stream in some way and then pass the modified data as output to the next transformation in the pipeline, the invention does not require these characteristics. According to the aspect, individual

transformations can receive input of expected form from more than one source or receive no input at all as would a transformation acting as a timestamp. According to the aspect, individual transformations, may not modify the data as would be encountered with a data store acting as a queue for downstream transformations.

[0108] According to the aspect, individual transformations may provide output to more than one downstream transformations. This ability lends itself to simulations where multiple possible choices might be made at a single step of a procedure all of which need to be analyzed. While only a single, simple use case has been offered for each example, in each case, that example was chosen for simplicity of description from a plurality of possibilities, the examples given should not be considered to limit the invention to only simplistic applications. Last, according to the invention, transformations in a transformation pipeline backbone may form a linear, a quasi-linear arrangement or may be cyclical, where the output of one of the internal transformations serves as the input of one of its antecedents allowing recursive analysis to be run. The result of transformation pipeline analysis may then be modified by results from batch analysis of the data stream and output **1606** in format predesigned by the authors of the analysis with could be human readable summary printout, human readable instruction printout, human-readable raw printout, data store, or machine encoded information of any format known to the art to be used in further automated analysis or action schema.

[0109] FIG. **17** is a process flow diagram of a method **1700** for an aspect of modeling the transformation pipeline module as a directed graph using graph theory. According to the aspect, the individual transformations **17102**, **17104**, **17106** of the transformation pipeline t.sub.1 . . . t.sub.n such that each t.sub.i T are represented as graph nodes. Transformations belonging to T are discrete transformations over individual datasets d.sub.i, consistent with classical functions. As such, each individual transformation t.sub.j, receives a set of inputs and produces a single output. The input of an individual transformation t.sub.i is defined with the function in: t.sub.i d.sub.1 . . . d.sub.k such that in(t.sub.i)={d.sub.1 . . . d.sub.k) and describes a transformation with k inputs. Similarly, the output of an individual transformation is defined as the function out: t.sub.i [ld.sub.1] to describe transformations that produce a single output (usable by other transformations). A dependency function can now be defined such that dep(t.sub.a,t.sub.b) out(t.sub.a)in(t.sub.b) The messages carrying the data stream through the transformation pipeline **1701**, **1703**, **1705** make up the graph edges. Using the above definitions, then, a transformation pipeline within the invention can be defined as G=(V,E) where message (t.sub.1, t.sub.2 . . . t.sub.(n-1), t.sub.n)V and all transformations t.sub.1 . . . t.sub.n and all dependencies dep(t.sub.i,t.sub.j)E **1707**.

[0110] FIG. **18** is a flow diagram illustrating an exemplary method for providing experience curation, according to an aspect of an embodiment. According to the aspect, the process begins at step **1801** when a distributed generative AI reasoning and action platform receives a user query directed to a generative AI system. The query may comprise a request for information, a summary, a request for a document, or some other action. The user may submit their query to the platform via an experience curation portal such as through a webapp or website accessed via an Internet browser operating on a computer (e.g., personal computer, laptop), or through an associated curation application which can be operated on a mobile computing device (e.g., smart phone, tablet, smart wearable, IoT device, etc.). In some implementations, the received user query may be sent to a data embedding system which can vectorize the query and store it in a vector database where it may be retrieved to be used as contextual data included in a query/prompt sent to a generative AI system.

[0111] At step **1802** the query is sent to the generative AI system which processes the query and returns a generated response which is received by the platform at step **1803**. At step **1804** curation system locates and retrieves any available user-defined rules or preferences. In some embodiments, the user-defined rules/preferences may be defined by an entity (e.g., a company). Exemplary rules or preferences can include, but are not limited to, conditional generation preferences, formatting rules, language rules, style rules, geographic rules, environmental rules, and timing rules. With

respect to conditional generation rules, the model can be conditioned on specific input data related to the individual, such as preferences, behavior, and characteristics. For example, in text generation, the model could be conditioned on a user's previous messages or writing style to generate more personalized responses. Formatting, style, and language rules are closely related and may be used to curate a response in a specific format (e.g., bullet points, paragraph, single sentence, numbered outline, CSV, etc.), response style (e.g., formal, informal, academic, accessible, abstract, casual, etc.), and the language in which a response is translated, respectively. At step **1805** curation system can curate the response based on the retrieved user-defined rules or preferences. For example, the system may filter out extraneous data, or personal information. As a last step **1806**, curation system returns the curated response to the user, thereby providing experience curation to a platform user.

[0112] FIG. **19** is a flow diagram illustrating an exemplary method for providing experience curation with using rich contextual data, according to an aspect of an embodiment. According to the aspect, the process begins at step **1901** when a distributed generative AI reasoning and action platform receives a user query directed to a generative AI system. The query may comprise a request for information, a summary, a request for a document, or some other action. The user may submit their query to the platform via an experience curation portal such as through a webapp or website accessed via an Internet browser operating on a computer (e.g., personal computer, laptop), or through an associated curation application which can be operated on a mobile computing device (e.g., smart phone, tablet, smart wearable, IoT device, etc.). In some implementations, the received user query may be sent to a data embedding system which can vectorize the query and store it in a vector database where it may be retrieved to be used as contextual data included in a query/prompt sent to an ML, AI, generative AI, planning, or automation/action orchestration system.

[0113] A DCG orchestrated model which employs a hierarchical classification and model selection regime for content (either in whole or in part) can enable much more accurate ultimate semantic performance. For example, a query/prompt can be submitted to the generative AI system with additional metadata associated with the context of the prompt itself as well as additional broader information about the user and the user's ongoing behavior and/or activities. At step **1902** the system obtains a plurality of rich context data associated with the user, the query, or both. A subset of the plurality of context data information may be obtained from a vector database, the vector database comprising a plurality of embedded contextual data. Embedded contextual data can comprise (but is not limited to) information obtained from an enterprise knowledge base and embedded queries/prompts. Context data associated with the user may comprise information obtained from or related to one or more of a computing device on which the user is accessing the curation system/platform, the geographic location the user is located, an action the user is performing during interaction with the curation system/platform, and timing data associated with the user, and/or the like. A subset of the plurality of obtained context data may be obtained from one or more marketplaces such as a data marketplace and/or an expert judgment marketplace. In some embodiments, the selection of context data may be based on one or more expert judgment scores assigned to an information source or dataset.

[0114] As an example, if a user is asking a generative AI enhanced search engine for “the best pizza” on her cell phone while driving at 55 mph on the road and not near her home (e.g. on vacation) this is massively different from the user being at home, on her couch, connected on her laptop, from her normal IP address, having just ran a series of searches for airline tickets to Italy and Neapolitan Pizza recipes. The additional device, user, recent behavior, etc. content can be used by a classifier alongside a prompt to help focus results on things that are not only relevant (e.g. pizza places near the user that are open now) but likely to be consistent with her broader needs/persona (e.g. if available, the suggestions could be looked at based on other budget, dining, etc. preferences like outdoor seating and meals below $20 per person). The same principle applies to more complicated and complex topics like medicine or finance or law.

[0115] At step **1903** the obtained plurality of context data may be processed into vectors by an embedding model and stored in the vector database.
[0116] At step **1904** the user query and the vectorized context data is sent to the generative AI system which processes the query and returns a generated response which accounts for the information contained in the vectorized context data and which is received by the platform at step **1905**. At step **1906** curation system locates and retrieves any available user-defined rules or preferences. In some embodiments, the user-defined rules/preferences may be defined by an entity (e.g., a company). Exemplary rules or preferences can include, but are not limited to, conditional generation preferences, formatting rules, language rules, style rules, geographic rules, environmental rules, and timing rules. With respect to conditional generation rules, the model can be conditioned on specific input data related to the individual, such as preferences, behavior, and characteristics. For example, in text generation, the model could be conditioned on a user's previous messages or writing style to generate more personalized responses. Formatting, style, and language rules are closely related and may be used to curate a response in a specific format (e.g., bullet points, paragraph, single sentence, numbered outline, CSV, etc.), response style (e.g., formal, informal, academic, accessible, abstract, casual, etc.), and the language in which a response is translated, respectively. At step **1907** curation system can curate the response based on the retrieved user-defined rules or preferences. For example, the system may filter out extraneous data, or personal information. As a last step **1908**, curation system returns the curated response to the user, thereby providing experience curation to a platform user.
[0117] FIG. **20** is a flow diagram illustrating an exemplary method for providing distributed neuro-symbolic reasoning and action model, according to an aspect of an embodiment. A neuro-symbolic model combines neural network-based approaches with symbolic reasoning to enable a more flexible and powerful reasoning system. In neuro-symbolic reasoning, neural networks are used to learn representations of data, similar to how they are used in deep learning. These learned representations can then be combined with symbolic representations and rules to perform reasoning tasks. This combination allows for the strengths of both approaches to be leveraged: the ability of neural networks to learn complex patterns from data, and the ability of symbolic reasoning to represent and manipulate abstract concepts and rules.
[0118] According to the aspect, the process begins at step **2001***a*-*c* wherein a plurality of input data is obtained from various sources. Examples of input data can include entity knowledge **2001***a*, context data **2001***b*, and expert knowledge **2001***c*. Other types of data may be obtained and may be dependent upon the embodiment and the particular use case. Data may be obtained from third-party services, entity databases/data warehouses/knowledge base and/or the like, and various marketplaces for data, algorithms, RAGs, and/or expert judgment. At step **2002** the obtained plurality of input data is vectorized using an embedding model and stored in a vector database. Vectorizing the data allows it to be used as input for processing by a neural network. At step **2003** platform **120** can train the neural network using the input data to learn patterns and relationships in the data. In some embodiments, this step may involve the use of labeled examples and supervised learning. A recurrent neural network or some other transformer-based model may be used as the basis for the neural network. At step **2004** the system maps the learned representations to symbolic concepts or rules. At this step, the system learns to represent the learned features or representations from the neural network in symbolic form. At step **2005** the system applies reasoning techniques to the symbolic representations to perform reasoning tasks. Examples of reasoning techniques that may be implemented can include, but are not limited to, logic rules or inference engines. This step may involve combining the learned representations with existing knowledge or rules to derive new conclusions. At this point in the process a feedback loop is created wherein feedback from the symbolic reasoning step is incorporated back into the neural network to refine the learned representations. This feedback loop helps to improve the performance of the system over time. As a last step **2006**, the trained, distributed GenAI reasoning and action model can generate output of the

reasoning process, which could be a decision, a prediction, or an action based on the input data and the reasoning process. In some embodiment, the input data may further include a query/prompt and metadata comprising various contextual information about the user and/or prompt.

[0119] FIG. **21** is a flow diagram illustrating an exemplary method for using a distributed computation graph system for creating structured representations or knowledge graphs from various data sources, and setting up a pipeline for continuous processing and monitoring of that data, according to an embodiment. According to the embodiment, the process begins at step **2101** when the platform receives a plurality of input data of interest from structured (e.g., databases, spreadsheets, etc.) and unstructured (e.g., documents, websites, social media, etc.) data sources. Data may be obtained using data extraction techniques such as, for example, web scraping, APIs, natural language processing, etc.). Additionally, or alternatively, the data may be obtained from a data marketplace (e.g., expert judgment, RAGs, models, datasets, etc.). At step **2102** platform creates a knowledge graph or structured representation from data of interest. This may comprise applying information extraction methods (e.g., named entity recognition, relation extraction, etc.) to extract entities and relationships from unstructured data and integrating the extracted information with structed data sources. Further, a knowledge graph representation can be built by creating nodes for entities and edges for relationships. Optionally, platform can be configured to create vector representations of entities/relationships using techniques like word embeddings.

[0120] At step **2103**, platform selects data of interest, knowledge graph of interest, vector database of interest, embedding of interest, model of interest or simulation of interest from marketplace and procures it. Platform can search/browse a marketplace or repository for relevant data sources, knowledge graphs, vector databases, models, simulations, etc., and evaluate the potential options based on factors like relevance, quality, and cost. This step may further include purchasing or licensing selected assets for use in a pipeline. As a next step **2104**, platform creates a new pipeline which will continue to process target of interest data on a continuous or periodic or aperiodic basis going forward. The pipeline may be designed (e.g., batch, streaming, etc.) based on the processing needs and can integrate components for data ingestion, knowledge graph updates, model execution and/or the like.

[0121] At step **2105**, platform can instantiate or reserve resources for the new pipeline based on estimated resource needs for storage, transport, compute, privacy, regulation/laws, safety, and prices for relevant services needed to handle future pipeline via a probabilistic representation of it. Platform may estimate pipeline resource requirements (storage, compute, networking, etc.) and also consider privacy, regulatory, and safety constraints that impact resource needs. Platform may use probabilistic modeling to forecast future resource demands. This step may further comprise provisioning cloud/on-premise resources (e.g., virtual machines, containers, databases, etc.) accordingly. As a last step **2106**, platform monitors and adjusts the pipeline going forward based on uncertainty quantification methods looking at model expectations versus empirical observations and expected future “exogenous” zone of interest. A pipeline may be monitored for performance, data drift, model accuracy, etc. and by tracking metrics like data volumes, processing times, error rates, and model accuracies. Platform may use techniques such as, for example, Bayesian modeling to quantify uncertainties in model parameters and propagate input/parameter uncertainties through the model(s) to get prediction uncertainties. Techniques such as bootstrap, cross-validation can also quantify model uncertainties. Platform can identify external variables (e.g., new regulations, market shifts, technology changes, etc.) that may impact the “zone of interest” and quantify potential impacts on pipeline performance/relevance. As an example, platform could implement monitoring for these variables using web scraping, news feeds, etc.

Exemplary Computing Environment

[0122] FIG. **22** illustrates an exemplary computing environment on which an embodiment described herein may be implemented, in full or in part. This exemplary computing environment describes computer-related components and processes supporting enabling disclosure of computer-

implemented embodiments. Inclusion in this exemplary computing environment of well-known processes and computer components, if any, is not a suggestion or admission that any embodiment is no more than an aggregation of such processes or components. Rather, implementation of an embodiment using processes and components described in this exemplary computing environment will involve programming or configuration of such processes and components resulting in a machine specially programmed or configured for such implementation. The exemplary computing environment described herein is only one example of such an environment and other configurations of the components and processes are possible, including other relationships between and among components, and/or absence of some processes or components described. Further, the exemplary computing environment described herein is not intended to suggest any limitation as to the scope of use or functionality of any embodiment implemented, in whole or in part, on components or processes described herein.

[0123] The exemplary computing environment described herein comprises a computing device **10** (further comprising a system bus **11**, one or more processors **20**, a system memory **30**, one or more interfaces **40**, one or more non-volatile data storage devices **50**), external peripherals and accessories **60**, external communication devices **70**, remote computing devices **80**, and cloud-based services **90**.

[0124] System bus **11** couples the various system components, coordinating operation of and data transmission between those various system components. System bus **11** represents one or more of any type or combination of types of wired or wireless bus structures including, but not limited to, memory busses or memory controllers, point-to-point connections, switching fabrics, peripheral busses, accelerated graphics ports, and local busses using any of a variety of bus architectures. By way of example, such architectures include, but are not limited to, Industry Standard Architecture (ISA) busses, Micro Channel Architecture (MCA) busses, Enhanced ISA (EISA) busses, Video Electronics Standards Association (VESA) local busses, a Peripheral Component Interconnects (PCI) busses also known as a Mezzanine busses, or any selection of, or combination of, such busses. Depending on the specific physical implementation, one or more of the processors **20**, system memory **30** and other components of the computing device **10** can be physically co-located or integrated into a single physical component, such as on a single chip. In such a case, some or all of system bus **11** can be electrical pathways within a single chip structure.

[0125] Computing device may further comprise externally-accessible data input and storage devices **12** such as compact disc read-only memory (CD-ROM) drives, digital versatile discs (DVD), or other optical disc storage for reading and/or writing optical discs **62**; magnetic cassettes, magnetic tape, magnetic disk storage, or other magnetic storage devices; or any other medium which can be used to store the desired content and which can be accessed by the computing device **10**. Computing device may further comprise externally-accessible data ports or connections **12** such as serial ports, parallel ports, universal serial bus (USB) ports, and infrared ports and/or transmitter/receivers. Computing device may further comprise hardware for wireless communication with external devices such as IEEE 1394 (“Firewire”) interfaces, IEEE 802.11 wireless interfaces, BLUETOOTH® wireless interfaces, and so forth. Such ports and interfaces may be used to connect any number of external peripherals and accessories **60** such as visual displays, monitors, and touch-sensitive screens **61**, USB solid state memory data storage drives (commonly known as “flash drives” or “thumb drives”) **63**, printers **64**, pointers and manipulators such as mice **65**, keyboards **66**, and other devices **67** such as joysticks and gaming pads, touchpads, additional displays and monitors, and external hard drives (whether solid state or disc-based), microphones, speakers, cameras, and optical scanners.

[0126] Processors **20** are logic circuitry capable of receiving programming instructions and processing (or executing) those instructions to perform computer operations such as retrieving data, storing data, and performing mathematical calculations. Processors **20** are not limited by the materials from which they are formed or the processing mechanisms employed therein, but are

typically comprised of semiconductor materials into which many transistors are formed together into logic gates on a chip (i.e., an integrated circuit or IC). The term processor includes any device capable of receiving and processing instructions including, but not limited to, processors operating on the basis of quantum computing, optical computing, mechanical computing (e.g., using nanotechnology entities to transfer data), and so forth. Depending on configuration, computing device **10** may comprise more than one processor. For example, computing device **10** may comprise one or more central processing units (CPUs) **21**, each of which itself has multiple processors or multiple processing cores, each capable of independently or semi-independently processing programming instructions. Further, computing device **10** may comprise one or more specialized processors such as a graphics processing unit (GPU) **22** configured to accelerate processing of computer graphics and images via a large array of specialized processing cores arranged in parallel.

[0127] System memory **30** is processor-accessible data storage in the form of volatile and/or nonvolatile memory. System memory **30** may be either or both of two types: non-volatile memory and volatile memory. Non-volatile memory **30***a* is not erased when power to the memory is removed, and includes memory types such as read only memory (ROM), electronically-erasable programmable memory (EEPROM), and rewritable solid state memory (commonly known as "flash memory"). Non-volatile memory **30***a* is typically used for long-term storage of a basic input/output system (BIOS) **31**, containing the basic instructions, typically loaded during computer startup, for transfer of information between components within computing device, or a unified extensible firmware interface (UEFI), which is a modern replacement for BIOS that supports larger hard drives, faster boot times, more security features, and provides native support for graphics and mouse cursors. Non-volatile memory **30***a* may also be used to store firmware comprising a complete operating system **35** and applications **36** for operating computer-controlled devices. The firmware approach is often used for purpose-specific computer-controlled devices such as appliances and Internet-of-Things (IoT) devices where processing power and data storage space is limited. Volatile memory **30***b* is erased when power to the memory is removed and is typically used for short-term storage of data for processing. Volatile memory **30***b* includes memory types such as random-access memory (RAM), and is normally the primary operating memory into which the operating system **35**, applications **36**, program modules **37**, and application data **38** are loaded for execution by processors **20**. Volatile memory **30***b* is generally faster than non-volatile memory **30***a* due to its electrical characteristics and is directly accessible to processors **20** for processing of instructions and data storage and retrieval. Volatile memory **30***b* may comprise one or more smaller cache memories which operate at a higher clock speed and are typically placed on the same IC as the processors to improve performance.

[0128] Interfaces **40** may include, but are not limited to, storage media interfaces **41**, network interfaces **42**, display interfaces **43**, and input/output interfaces **44**. Storage media interface **41** provides the necessary hardware interface for loading data from non-volatile data storage devices **50** into system memory **30** and storage data from system memory **30** to non-volatile data storage device **50**. Network interface **42** provides the necessary hardware interface for computing device **10** to communicate with remote computing devices **80** and cloud-based services **90** via one or more external communication devices **70**. Display interface **43** allows for connection of displays **61**, monitors, touchscreens, and other visual input/output devices. Display interface **43** may include a graphics card for processing graphics-intensive calculations and for handling demanding display requirements. Typically, a graphics card includes a graphics processing unit (GPU) and video RAM (VRAM) to accelerate display of graphics. One or more input/output (I/O) interfaces **44** provide the necessary support for communications between computing device **10** and any external peripherals and accessories **60**. For wireless communications, the necessary radio-frequency hardware and firmware may be connected to I/O interface **44** or may be integrated into I/O interface **44**.

[0129] Non-volatile data storage devices **50** are typically used for long-term storage of data. Data

on non-volatile data storage devices **50** is not erased when power to the non-volatile data storage devices **50** is removed. Non-volatile data storage devices **50** may be implemented using any technology for non-volatile storage of content including, but not limited to, CD-ROM drives, digital versatile discs (DVD), or other optical disc storage; magnetic cassettes, magnetic tape, magnetic disc storage, or other magnetic storage devices; solid state memory technologies such as EEPROM or flash memory; or other memory technology or any other medium which can be used to store data without requiring power to retain the data after it is written. Non-volatile data storage devices **50** may be non-removable from computing device **10** as in the case of internal hard drives, removable from computing device **10** as in the case of external USB hard drives, or a combination thereof, but computing device will typically comprise one or more internal, non-removable hard drives using either magnetic disc or solid state memory technology. Non-volatile data storage devices **50** may store any type of data including, but not limited to, an operating system **51** for providing low-level and mid-level functionality of computing device **10**, applications **52** for providing high-level functionality of computing device **10**, program modules **53** such as containerized programs or applications, or other modular content or modular programming, application data **54**, and databases **55** such as relational databases, non-relational databases, object oriented databases, BOSQL databases, and graph databases.

[0130] Applications (also known as computer software or software applications) are sets of programming instructions designed to perform specific tasks or provide specific functionality on a computer or other computing devices. Applications are typically written in high-level programming languages such as C++, Java, and Python, which are then either interpreted at runtime or compiled into low-level, binary, processor-executable instructions operable on processors **20**. Applications may be containerized so that they can be run on any computer hardware running any known operating system. Containerization of computer software is a method of packaging and deploying applications along with their operating system dependencies into self-contained, isolated units known as containers. Containers provide a lightweight and consistent runtime environment that allows applications to run reliably across different computing environments, such as development, testing, and production systems.

[0131] The memories and non-volatile data storage devices described herein do not include communication media. Communication media are means of transmission of information such as modulated electromagnetic waves or modulated data signals configured to transmit, not store, information. By way of example, and not limitation, communication media includes wired communications such as sound signals transmitted to a speaker via a speaker wire, and wireless communications such as acoustic waves, radio frequency (RF) transmissions, infrared emissions, and other wireless media.

[0132] External communication devices **70** are devices that facilitate communications between computing device and either remote computing devices **80**, or cloud-based services **90**, or both. External communication devices **70** include, but are not limited to, data modems **71** which facilitate data transmission between computing device and the Internet **75** via a common carrier such as a telephone company or internet service provider (ISP), routers **72** which facilitate data transmission between computing device and other devices, and switches **73** which provide direct data communications between devices on a network. Here, modem **71** is shown connecting computing device **10** to both remote computing devices **80** and cloud-based services **90** via the Internet **75**. While modem **71**, router **72**, and switch **73** are shown here as being connected to network interface **42**, many different network configurations using external communication devices **70** are possible. Using external communication devices **70**, networks may be configured as local area networks (LANs) for a single location, building, or campus, wide area networks (WANs) comprising data networks that extend over a larger geographical area, and virtual private networks (VPNs) which can be of any size but connect computers via encrypted communications over public networks such as the Internet **75**. As just one exemplary network configuration, network interface **42** may be

connected to switch **73** which is connected to router **72** which is connected to modem **71** which provides access for computing device **10** to the Internet **75**. Further, any combination of wired **77** or wireless **76** communications between and among computing device **10**, external communication devices **70**, remote computing devices **80**, and cloud-based services **90** may be used. Remote computing devices **80**, for example, may communicate with computing device through a variety of communication channels **74** such as through switch **73** via a wired **77** connection, through router **72** via a wireless connection **76**, or through modem **71** via the Internet **75**. Furthermore, while not shown here, other hardware that is specifically designed for servers may be employed. For example, secure socket layer (SSL) acceleration cards can be used to offload SSL encryption computations, and transmission control protocol/internet protocol (TCP/IP) offload hardware and/or packet classifiers on network interfaces **42** may be installed and used at server devices.

[0133] In a networked environment, certain components of computing device **10** may be fully or partially implemented on remote computing devices **80** or cloud-based services **90**. Data stored in non-volatile data storage device **50** may be received from, shared with, duplicated on, or offloaded to a non-volatile data storage device on one or more remote computing devices **80** or in a cloud computing service **92**. Processing by processors **20** may be received from, shared with, duplicated on, or offloaded to processors of one or more remote computing devices **80** or in a distributed computing service **93**. By way of example, data may reside on a cloud computing service **92**, but may be usable or otherwise accessible for use by computing device **10**. Also, certain processing subtasks may be sent to a microservice **91** for processing with the result being transmitted to computing device **10** for incorporation into a larger processing task. Also, while components and processes of the exemplary computing environment are illustrated herein as discrete units (e.g., OS **51** being stored on non-volatile data storage device **51** and loaded into system memory **35** for use) such processes and components may reside or be processed at various times in different components of computing device **10**, remote computing devices **80**, and/or cloud-based services **90**.

[0134] In an implementation, the disclosed systems and methods may utilize, at least in part, containerization techniques to execute one or more processes and/or steps disclosed herein. Containerization is a lightweight and efficient virtualization technique that allows you to package and run applications and their dependencies in isolated environments called containers. One of the most popular containerization platforms is Docker, which is widely used in software development and deployment. Containerization, particularly with open-source technologies like Docker and container orchestration systems like Kubernetes, is a common approach for deploying and managing applications. Containers are created from images, which are lightweight, standalone, and executable packages that include application code, libraries, dependencies, and runtime. Images are often built from a Dockerfile or similar, which contains instructions for assembling the image. Dockerfiles are configuration files that specify how to build a Docker image. Systems like Kubernetes also support containerd or CRI-O. They include commands for installing dependencies, copying files, setting environment variables, and defining runtime configurations. Docker images are stored in repositories, which can be public or private. Docker Hub is an exemplary public registry, and organizations often set up private registries for security and version control using tools such as Hub, JFrog Artifactory and Bintray, Github Packages or Container registries. Containers can communicate with each other and the external world through networking. Docker provides a bridge network by default, but can be used with custom networks. Containers within the same network can communicate using container names or IP addresses.

[0135] Remote computing devices **80** are any computing devices not part of computing device **10**. Remote computing devices **80** include, but are not limited to, personal computers, server computers, thin clients, thick clients, personal digital assistants (PDAs), mobile telephones, watches, tablet computers, laptop computers, multiprocessor systems, microprocessor based systems, set-top boxes, programmable consumer electronics, video game machines, game consoles,

portable or handheld gaming units, network terminals, desktop personal computers (PCs), minicomputers, main frame computers, network nodes, virtual reality or augmented reality devices and wearables, and distributed or multi-processing computing environments. While remote computing devices **80** are shown for clarity as being separate from cloud-based services **90**, cloud-based services **90** are implemented on collections of networked remote computing devices **80**.

[0136] Cloud-based services **90** are Internet-accessible services implemented on collections of networked remote computing devices **80**. Cloud-based services are typically accessed via application programming interfaces (APIs) which are software interfaces which provide access to computing services within the cloud-based service via API calls, which are pre-defined protocols for requesting a computing service and receiving the results of that computing service. While cloud-based services may comprise any type of computer processing or storage, three common categories of cloud-based services **90** are microservices **91**, cloud computing services **92**, and distributed computing services **93**.

[0137] Microservices **91** are collections of small, loosely coupled, and independently deployable computing services. Each microservice represents a specific computing functionality and runs as a separate process or container. Microservices promote the decomposition of complex applications into smaller, manageable services that can be developed, deployed, and scaled independently. These services communicate with each other through well-defined application programming interfaces (APIs), typically using lightweight protocols like HTTP, gRPC, or message queues such as Kafka. Microservices **91** can be combined to perform more complex processing tasks.

[0138] Cloud computing services **92** are delivery of computing resources and services over the Internet **75** from a remote location. Cloud computing services **92** provide additional computer hardware and storage on as-needed or subscription basis. Cloud computing services **92** can provide large amounts of scalable data storage, access to sophisticated software and powerful server-based processing, or entire computing infrastructures and platforms. For example, cloud computing services can provide virtualized computing resources such as virtual machines, storage, and networks, platforms for developing, running, and managing applications without the complexity of infrastructure management, and complete software applications over the Internet on a subscription basis.

[0139] Distributed computing services **93** provide large-scale processing using multiple interconnected computers or nodes to solve computational problems or perform tasks collectively. In distributed computing, the processing and storage capabilities of multiple machines are leveraged to work together as a unified system. Distributed computing services are designed to address problems that cannot be efficiently solved by a single computer or that require large-scale computational power. These services enable parallel processing, fault tolerance, and scalability by distributing tasks across multiple nodes.

[0140] Although described above as a physical device, computing device **10** can be a virtual computing device, in which case the functionality of the physical components herein described, such as processors **20**, system memory **30**, network interfaces **40**, and other like components can be provided by computer-executable instructions. Such computer-executable instructions can execute on a single physical computing device, or can be distributed across multiple physical computing devices, including being distributed across multiple physical computing devices in a dynamic manner such that the specific, physical computing devices hosting such computer-executable instructions can dynamically change over time depending upon need and availability. In the situation where computing device **10** is a virtualized device, the underlying physical computing devices hosting such a virtualized computing device can, themselves, comprise physical components analogous to those described above, and operating in a like manner. Furthermore, virtual computing devices can be utilized in multiple layers with one virtual computing device executing within the construct of another virtual computing device. Thus, computing device **10** may be either a physical computing device or a virtualized computing device within which

computer-executable instructions can be executed in a manner consistent with their execution by a physical computing device. Similarly, terms referring to physical components of the computing device, as utilized herein, mean either those physical components or virtualizations thereof performing the same or equivalent functions.
[0141] The skilled person will be aware of a range of possible modifications of the various aspects described above. Accordingly, the present invention is defined by the claims and their equivalents.

Claims

1. A computing system comprising: at least a memory and one or more hardware processors configured to: obtain a plurality of input data, the input data comprising enterprise knowledge and expert knowledge; process the input data using an embedding engine to generate a vectorized dataset; apply a machine learning model to the vectorized dataset, the machine learning model having been trained to identify relationships among the input data, to generate a parameter set; map the parameter set to a plurality of symbolic rules; generate a structured execution graph representing a course of action by applying a symbolic reasoning engine to the symbolic rules; determine, for at least a subset of nodes in the structured execution graph, one or more physical computing locations for execution; and initiate execution of the structured execution graph at the one or more determined physical computing locations.
2. The computing system of claim 1, further comprising a model interaction interface configured to: automatically generate a natural language query based on at least a portion of the symbolic rules; transmit the query to a large language model; and receive a response for modifying, supplementing, or validating one or more nodes of the structured execution graph.
3. The computing system of claim 1, wherein the symbolic reasoning engine incorporates retrieval-augmented generation (RAG) information into the execution of one or more of the symbolic rules.
4. The computing system of claim 1, wherein the symbolic reasoning engine generates the structured execution graph based in part on user-specific information derived from enterprise knowledge or prior interactions.
5. The computing system of claim 1, wherein determining the physical computing locations comprises evaluating device-specific locality information associated with a user device.
6. The computing system of claim 1, wherein the expert knowledge comprises scored datasets or scored model output.
7. The computing system of claim 6, wherein the expert knowledge is obtained from an expert knowledge marketplace.
8. A computer-implemented method comprising the steps of: obtaining a plurality of input data, the input data comprising enterprise knowledge and expert knowledge; processing the obtained plurality of input data using an embedding model to create a vectorized dataset; obtaining a plurality of input data, the input data comprising enterprise knowledge and expert knowledge; processing the input data using an embedding engine to generate a vectorized dataset; applying a machine learning model to the vectorized dataset, the machine learning model having been trained to identify relationships among the input data, to generate a parameter set; mapping the parameter set to a plurality of symbolic rules; generate a structured execution graph representing a course of action by applying a symbolic reasoning engine to the plurality of symbolic rules; determining, for at least a subset of nodes of the structured execution graph, one or more physical computing locations for execution based on locality constraints, regulatory limitations, or available resources; and initiating execution of the structured execution graph at the one or more determined physical computing locations.
9. The computer-implemented method of claim 8, further comprising: generating a natural language query based on at least a portion of the symbolic rules; transmitting the query to a large language model; and receiving a response for modifying, supplementing, or validating one or more

nodes of the structured execution graph.
10. The computer-implemented method of claim **98**, wherein the symbolic reasoning engine incorporates retrieval-augmented generation (RAG) information to inform one or more symbolic rules.
11. The computer-implemented method of claim 8, wherein the symbolic reasoning engine accounts for information associated with an action a user of the user device is performing during interaction with the platform.
12. The computer-implemented method of claim 8 wherein determining the physical computing locations comprises evaluating device-specific locality information associated with a user device.
13. The computer-implemented method of claim 8, wherein the expert knowledge comprises scored datasets or scored model output.
14. The computer-implemented method of claim 13, wherein the expert knowledge is obtained from an expert knowledge marketplace.
15. A system comprising one or more computers each with a memory and at least one processor and executable instructions that, when executed on one or more of the computers, cause the system to: obtain a plurality of input data, the input data comprising enterprise knowledge and expert knowledge; process the obtained plurality of input data using an embedding model to create a vectorized dataset; apply a machine learning model to the vectorized dataset, the machine learning model having been trained to identify relationships among the input data, to generate a parameter set; map the parameter set to a plurality of symbolic rules; generate a structured execution graph representing a course of action by applying a symbolic reasoning engine to the plurality of symbolic rules; determine, for at least a subset of nodes in the structured execution graph, one or more physical computing locations for execution based on locality constraints, regulatory limitations, or available resources; and initiate execution of the structured execution graph at the one or more determined physical computing locations.
16. The system of claim 15, further comprising a model interaction interface configured to: generate a natural language query based on at least a portion of the symbolic rules; transmit the query to a large language model; and receive a response for modifying, supplementing, or validating one or more nodes of the structured execution graph.
17. The system of claim 15, wherein the symbolic reasoning engine incorporates retrieval-augmented generation (RAG) information to inform one or more symbolic rules.
18. The system of claim 15, wherein the symbolic reasoning engine accounts for information associated with an action a user of the user device is performing during interaction with the platform.
19. The system of claim 15, wherein determining the physical computing locations comprises evaluating device-specific locality information associated with a user device.
20. The system of claim 15, wherein the expert knowledge comprises scored datasets or scored model output.
21. The system of claim 20, wherein the expert knowledge is obtained from an expert knowledge marketplace.
22. Non-transitory, computer-readable storage media having computer executable instructions embodied thereon that, when executed by one or more processors of a computing system, cause the computing system to: obtain a plurality of input data, the input data comprising enterprise knowledge and expert knowledge; process the obtained plurality of input data using an embedding model to create a vectorized dataset; apply a machine learning model to the vectorized dataset, the machine learning model having been trained to identify relationships among the input data, to generate a parameter set; map the parameter set to a plurality of symbolic rules; generate a structured execution graph representing a course of action by applying a symbolic reasoning engine to the symbolic rules; determine, for at least a subset of nodes in the structured execution graph, one or more physical computing locations for execution based on locality constraints, regulatory

limitations, or available resources; and initiate execution of the structured execution graph at the one or more determined physical computing locations.

23. The system of claim 22, further comprising a model interaction interface configured to: generate a natural language query based on at least a portion of the symbolic rules; transmit the query to a large language model; and receive a response for modifying, supplementing, or validating one or more nodes of the structured execution graph.

24. The system of claim 22, wherein the symbolic reasoning engine incorporates retrieval augmented generation (RAG) information to inform one or more symbolic rules.

25. The system of claim 22, wherein the symbolic reasoning engine accounts for information associated with an action a user of the user device is performing during interaction with the platform.

26. The system of claim 22, wherein determining the physical computing locations comprises evaluating device-specific locality information associated with a user device.

27. The system of claim 22, wherein the expert knowledge comprises scored datasets or scored model output.

28. The system of claim 27, wherein the expert knowledge is obtained from an expert knowledge marketplace.