

System and Method for Processing Language Model Outputs Using Quantum Computing

Technical Field

The present application relates generally to data processing techniques, and in particular systems and methods for processing language model outputs using quantum computing.

5 Background

Recent advancements in artificial intelligence have facilitated the development of sophisticated large language models, which demonstrate capabilities for generating text with notable fluency and contextual accuracy. These models, such as GPT-3 and Claude, have expanded the scope of natural language processing (NLP) applications, enabling
10 functionalities that range from automated content generation to complex conversational interfaces. However, despite these advancements, the extensive data produced by such models presents considerable challenges in processing and analysis.

The outputs of large language models are typically unstructured and voluminous, containing complex semantic dependencies that are not easily deciphered using traditional data
15 processing methodologies. Furthermore, the absence of a standardized, machine-readable format for these outputs impedes their integration with downstream systems, particularly those that leverage quantum computing capabilities to address classical computational constraints.

Moreover, the high volume and dimensionality of the data generated by these language
20 models pose computational challenges. Traditional NLP tasks such as semantic similarity assessment, sentiment classification, and entity recognition require increasingly more resources as the amount of data grows. Current algorithms and computational frameworks, based on classical computing paradigms, struggle to efficiently handle the rising volume and complexity of data.

Given the limitations of classical computing, the integration of quantum computing into the
25 processing of language model outputs presents a promising approach, offering the potential to perform certain computations substantially faster than classical computers. However, integrating classical NLP systems with quantum computing infrastructure encounters several obstacles. Robust mechanisms for data exchange and synchronization between classical
30 and quantum computing paradigms are notably lacking, which limits the potential for leveraging their combined strengths in language understanding and generation tasks. Additionally, most existing quantum computing tools and libraries are designed primarily for structured, numerical datasets and do not adequately support the unstructured text outputs from large language models.

Therefore, it is desirable to provide a system and method capable of processing large amounts of unstructured textual data with improved efficiency to address the disadvantages or limitations of the existing technologies or, at the very least, provide the public with a useful alternative.

5 Summary

Embodiments herein provide new and useful systems and methods for processing outputs from language models, implementing quantum natural language processing, and interfacing between classical computing infrastructures and quantum computing environments.

10 In broad terms, the present disclosure proposes a system for data processing, including a classical data interface configured to receive data from a classical computing environment, a data encoding module configured to encode the received data into a format compatible with a quantum computing environment, a quantum processing module within the quantum computing environment configured to process the encoded data, a communication interface configured to transfer the encoded data to the quantum computing environment and retrieve
15 the processed data from the quantum computing environment, and a data decoding module configured to decode the processed data into a format compatible with the classical computing environment.

In embodiments, the data encoding module is further configured to transform the received data into a structured dataset that is compatible with a metadata standard.

20 In embodiments, the structured dataset that is compatible with a metadata standard corresponds to an ISO 11179 compliant JSON representation.

In implementations, the quantum processing module is configured to use at least one quantum algorithm selected from the group comprising of quantum support vector machines (QSVMs) and quantum neural networks (QNNs).

25 In implementations, a task scheduler is provided for allocating tasks between the classical computing environment and quantum computing environment based on at least one of the following criteria: complexity of the data, suitability of processing requirements for quantum or classical computation, and available computational capacity in both computing environments.

30 The present disclosure further proposes a method for data processing, including receiving data from a classical computing environment. The method may further include encoding the received data into a format compatible with a quantum computing environment using a data encoding module, transferring the encoded data to the quantum computing environment,

processing the encoded data in the quantum computing environment using a quantum processor, retrieving the processed data from the quantum computing environment, and decoding the processed data into a format compatible with the classical computing environment using a data decoding module.

- 5 In implementations, the decoded processed data is further integrated into downstream applications in the classical computing environment.

In implementations, the method further includes applying a plurality of task scheduling algorithms within the quantum computing environment based on characteristics of quantum processors.

- 10 In embodiments, the encoded data transferred to the quantum computing environment and the processed data decoded for the classical computing environment are both in a structured data format that is compatible with a metadata standard.

- In embodiments, the method is implemented as part of a hybrid quantum-classical computing system that supports a hardware-agnostic abstraction layer which allows the
15 computing system to interact with a plurality of quantum processors.

- The present disclosure further proposes a method for transforming language model outputs, including parsing an output dataset corresponding to one or more language models to extract a dataset, processing the extracted dataset to generate a refined dataset, encoding the refined dataset into a structured dataset including metadata parameters, wherein the
20 structured dataset is configured for quantum processing, and evaluating the structured dataset for compatibility with a plurality of quantum computing systems.

In embodiments, the metadata parameters comprise at least one of qubit requirements, quantum gate operations, and connectivity between qubits and a quantum processing unit.

- In implementations, the structured dataset is configured for quantum processing
25 corresponds to an international protocol specific to quantum computing, high-level petri nets, or information technology.

In embodiments, the structured dataset is ISO 11179 compliant for providing compatibility with a plurality of quantum computing systems.

- The present disclosure further proposes a method for processing language model outputs,
30 including analyzing output data from one or more language models to extract a dataset, preparing the extracted dataset for compatibility with a standardized data format and quantum computing, encoding the prepared dataset into a structured dataset, wherein the

structured dataset includes metadata for quantum computing operations, and evaluating the structured dataset for compatibility with a metadata standard.

The present disclosure further proposes a system for quantum natural language processing, including a quantum computing device configured to execute quantum circuits for processing quantum states, and a memory storing instructions executable by a processor to encode input data into the quantum states using quantum circuits executed by the quantum computing device. The memory for further store instructions executable by a processor to apply one or more quantum operations via the quantum circuits to the encoded quantum states for processing at least one natural language task selected from the group comprising of semantic analysis, sentiment classification, named entity recognition, and text summarization, measure the encoded quantum states after the application of the one or more quantum operations to determine a result of the processing of the at least one natural language task, and output the determined result in a structured dataset format suitable for classical data processing.

In embodiments, the quantum operations include a Quantum Latent Semantic Analysis (QLSA) to identify semantic relationships within the text data.

In implementations, the quantum operations include at least one of a Quantum Support Vector Machine (QSVM), a Quantum Generative Adversarial Network (QGAN), or a Quantum Variational Autoencoder (QVAE).

In implementations, the quantum operations include at least one of quantum entanglement and superposition principles to enhance processing speed for the at least one natural language task.

In embodiments, the structured dataset format is ISO 11179 compliant for providing compatibility with a plurality of quantum computing systems.

In embodiments, the input data is provided as a structured dataset that is compatible with a metadata standard prior to encoding of the input data.

In implementations, the quantum operations include a quantum text summarization algorithm configured to generate summaries of text by using at least one of quantum phase estimation and quantum amplitude amplification.

In embodiments, the structured dataset format includes metadata corresponding to at least one of a natural language task, a quantum operation applied, a device parameter, and a number of quantum measurements.

In embodiments, the quantum operations include the use of Hadamard and rotation gates within the quantum circuits to encode a plurality of sentiment features into the quantum states.

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

Brief Description of the Drawings:

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

FIG. 1 is a functional block diagram of an example of a quantum-classical integration system for processing language model outputs, according to an embodiment herein.

FIG. 2 is a flowchart illustrating an example high-level process of using the system of FIG. 1 for transforming and processing language data through quantum computations, according to an embodiment herein.

FIG. 3 is a flowchart illustrating an example high-level process of using a quantum computing system for processing natural language tasks, according to an embodiment herein.

FIG. 4 is a flowchart illustrating an example process implemented by a quantum data representation module for preparing language model outputs for quantum computing, according to an embodiment herein.

FIG. 5 is a block diagram illustrating an example quantum computing framework for processing unstructured input data from language models through a series of quantum and classical modules, according to an embodiment herein.

FIG. 6 is a block diagram illustrating an example computer system which may be configured to implement the systems and methods as disclosed herein.

Detailed Description

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

should not be construed as limited to the embodiments set forth herein, shown in the FIGs, and/or described below.

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

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

The present disclosure provides a quantum computing framework designed for handling extensive textual outputs from advanced large language models like Claude and GPT. This framework combines quantum algorithms, standardized data representations, and a combination of classical and quantum computing techniques to address the challenges presented by the vast volume and complexity of unstructured language data.

Furthermore, the disclosure describes systems and methods for converting raw, unstructured text from language models into a structured, machine-readable format that adheres to a metadata registry standard such as ISO 11179. This conversion process may involve utilizing a JSON structure with metadata fields specific to quantum computing, allowing for integration with quantum computing systems and algorithms. By converting language model outputs into a format optimized for quantum computing, the quantum computing framework establishes a foundation for efficient and scalable processing of textual data.

The quantum computing framework includes three primary components: a Quantum-Classical Integration Module, a Quantum Processing Module, and a Quantum Data Representation Module. These components collaborate to significantly improve natural language processing and generation capabilities of the computing framework.

Quantum-Classical Integration Module

FIG. 1 is a functional block diagram of an example of a quantum-classical integration system 100 for processing language model outputs, according to an embodiment herein. This diagram illustrates components of the system 100 including a Classical Data Interface 102, Data Encoding Module 104, Quantum Processing Module 106, and Data Decoding Module 108, which collectively enable the transformation and processing of data between classical and quantum computing systems.

FIG. 2 is a flowchart 200 illustrating an example high-level process of using the system depicted in FIG. 1 for transforming and processing language data through quantum computations, according to an embodiment herein. The flowchart outlines a sequence from receiving to decoding data, thus enabling quantum processing starting in a hybrid-classical infrastructure.

In step 202, data is received from a classical computing environment. This initial step involves the collection of outputs from language models operating within classical systems, thereby initiating the integration of data into the quantum system.

In step 204, the received data is encoded by the Data Encoding Module 104. This encoding process transforms data into a format compatible with a quantum computing environment, preparing it for subsequent quantum processing.

In embodiments, the encoding module 104 of the Quantum-Classical Integration Module 100 may use various data encoding techniques to transfer classical language data onto quantum states, enabling efficient processing through quantum algorithms. In example embodiments, the encoding techniques employed by the encoding module 104 may include:

Amplitude Encoding: Amplitude encoding may include assigning each language token—whether a word, character, or subword unit—to a basis state of a qubit register. The probability amplitudes of these basis states may represent the embedding vector of the token.

Specialized data preparation circuits may be used to load these normalized embedding vectors into the quantum state amplitudes. This method of encoding provides for exponentially compact storage and parallel processing of language data, enhancing efficiency. For instance, the word "quantum" could be encoded as a superposition state: $\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$, where the amplitudes α , β , γ , and δ capture its semantic vector.

Angle Encoding: In angle encoding, language features may be encoded through the rotation angles of single-qubit gates that are applied to the quantum state. Each token's continuous-valued embedding vector may be discretized into a sequence of rotation angles, which can be implemented using Pauli rotations (RX, RY, RZ) or general unitaries (U3). The angle encoding preserves the geometric relationships between language embeddings, which helps capture

linguistic nuances. For example, the sentiment score of a word, which ranges from $[-1, +1]$, could be encoded as an $RY(\theta)$ rotation, where θ is calculated as $\pi (\text{score} + 1)/2$.

Tensor Product Encoding: Tensor product encoding may be utilized for structured language data, such as parse trees or knowledge graphs. In tensor product encoding, each node in the structure may be mapped to a qubit, with edges represented through the entanglement between these qubits. This method naturally represents the compositional and relational aspects of language, effectively mirroring the underlying linguistic structure. An example of this would be encoding a dependency parse tree as a tensor product of subtree states: $|\text{noun_phrase}\rangle \otimes |\text{verb_phrase}\rangle$.

Quantum Dictionary Encoding: Quantum dictionary encoding can make use of a quantum associative memory (QAM) to store language tokens and their corresponding quantum states. The QAM may be constructed as a superposition of key-value pairs, where the keys are classical token IDs, and the values are quantum embedding states. Quantum search algorithms, such as Grover's algorithm, can then efficiently retrieve the quantum embedding for a specific token. For example, the QAM might store $|\text{word1_id}\rangle|\text{word1_embedding}\rangle + |\text{word2_id}\rangle|\text{word2_embedding}\rangle + \dots$, facilitating rapid quantum embedding lookup.

Quantum Signal Processing Encoding: Quantum signal processing encoding may be applied to continuous language signals like audio or time-series data. In this approach, the signal may be discretized and mapped onto the amplitudes and phases of a quantum state using quantum Fourier transform (QFT) circuits. Techniques such as quantum phase estimation and amplitude estimation are then used to extract relevant signal features and properties. An example of Quantum Signal Processing Encoding is encoding a speech signal as a quantum state $|\psi\rangle = \sum_k \alpha_k |k\rangle$, where the amplitude α_k is proportional to the signal frequencies.

After encoding, the system 100 may perform an evaluation to ensure that the structured data adheres to established metadata standards, such as ISO 11179. This standard facilitates the structuring of dataset metadata, ensuring that data elements are not only correctly formatted but also accompanied by comprehensive metadata that enhances its usability in quantum processing. The system 100 may be configured with validation mechanisms that routinely assess the structured data against these metadata requirements, ensuring data transferred to the quantum processing module 106 complies with these standards. Additionally, the system's 100 capability to manage and dynamically adapt to updates in established metadata standards, such as ISO 11179 facilitates ongoing compliance and ensures interoperability of data across various computing platforms.

In order to facilitate robust and secure data transmission between the classical computing environment and the quantum computing environment, the system 100 may include a communication interface. This communication interface may use advanced data transmission protocols configured to ensure high-speed data transfer while maintaining data integrity and security. The communication protocol may incorporate features such as end-to-end encryption and real-time error correction to safeguard data against unauthorized access and transmission errors. Furthermore, the interface may be equipped with dynamic bandwidth allocation capabilities to efficiently manage the voluminous data flow inherent to processed language model outputs, ensuring that data latency is minimized and throughput maximized. In example embodiments, the communication interface is configured to transfer the encoded data to the quantum computing environment and retrieve the processed data from the quantum computing environment.

Data Transfer and Quantum Processing:

Following the encoding process, in step 206, the encoded data is transferred to the Quantum Processing Module 106. This step facilitates the transfer of the prepared data into the quantum computing environment for further processing.

In step 208, the encoded data is processed using quantum operations within the Quantum Processing Module 106. These operations utilize quantum mechanical properties to analyse and process the data, in order to achieve more efficient results than what is achievable by classical systems alone. Within the quantum processing environment, the quantum processing module 106, configured to process the encoded data, utilizes specific quantum algorithms, including quantum support vector machines (QSVMs) and quantum neural networks (QNNs), enhancing data processing efficiency and accuracy.

Decoding and Data Reintegration:

Finally, in step 210, the processed data is retrieved and decoded back into a format that is compatible with the classical computing environment. The Data Decoding Module 108 decodes the processed data back into a format compatible with the classical computing environment, ensuring that the structured data retains its integrity and usability within conventional systems. In other words, the decoded data is prepared to be re-integrated into the classical computing environment for further use or analysis. Upon successful decoding, the processed data may be integrated into downstream applications within the classical computing environment through a sophisticated API framework. This framework supports various data formats and standards, such as JSON or XML, enabling decoded data to be directly utilized by business intelligence systems, real-time monitoring applications, and advanced analytics platforms.

Decoding quantum states back into classical language data via data decoding module 108 may involve the use of quantum state tomography, amplitude estimation, error mitigation, and hybrid quantum-classical approaches. The choice of the decoding technique used by the data decoding module 108 can depend on factors such as the specific encoding method, language data structure, and quantum algorithm being employed. Designing an effective decoding methodology helps minimize the impact of quantum measurement errors and accurately retrieve classical language information from the quantum-processed states. Some considerations and techniques for decoding the quantum states may include:

Quantum State Tomography: Quantum state tomography may include performing multiple measurements on identically prepared copies of a quantum state to reconstruct its full configuration. Quantum state tomography may use techniques such as maximum likelihood estimation (MLE) to estimate the state's amplitudes and phases based on the outcomes of these measurements.

Quantum Amplitude Estimation: Quantum amplitude estimation (QAE) may be employed when the squared amplitudes of basis states represent probabilities of language data in amplitude-encoded states. Utilizing algorithms such as quantum phase estimation and quantum counting, QAE can efficiently estimate these probabilities. QAE offers quadratic speedups compared to classical sampling methods, which significantly enhances the efficiency of language data retrieval.

Quantum Measurement Error Mitigation: Quantum measurements may be inherently susceptible to readout errors resulting from hardware imperfections and environmental noise. To address this, error mitigation techniques are applied to the measurement outcomes, including readout error correction and post-processing noise filters. These error mitigation processes are designed to enhance the fidelity of the decoded classical language data, ensuring more reliable and accurate results.

Quantum-Classical Hybrid Decoding: In processing complex and intricate language data structures such as graphs or sequences, hybrid quantum-classical decoding strategies may be employed. These decoding strategies use quantum circuits that extract relevant features or properties from the encoded quantum state. Following the quantum processing, classical post-processing algorithms—such as clustering, parsing, or inference—may be applied to the measured quantum features to reconstruct and interpret the language data effectively. This hybrid approach leverages the strengths of both quantum and classical computing paradigms to achieve optimal decoding results.

To achieve optimal performance and scalability, the Quantum-Classical Integration Module 100 may incorporate intelligent workload distribution mechanisms. In this context, a task scheduler may be configured to assess and allocate tasks intelligently between the classical and quantum computing environments. The task scheduler evaluates the complexity of the incoming data, the suitability of available algorithms for either quantum or classical processing, and the current computational capacity of both systems. By doing so, it can determine the most efficient processing route for each task, whether that involves direct classical computing methods, quantum computing techniques, or a hybrid approach that leverages the strengths of both. This ensures that each task is processed in the most appropriate computational environment, optimizing resource usage and minimizing process latency.

Furthermore, the task scheduler may continuously monitor system performance, adapting task allocation in real-time to accommodate changes in data complexity and system availability, thereby maintaining optimal system performance throughout operations. To further enhance its operational efficiency, the system 100 may incorporate a task scheduling framework within the quantum computing environment. This framework may comprise multiple task scheduling algorithms, each designed to optimally utilize the unique characteristics and capabilities of the quantum processors in use for quantum computing. These task scheduling algorithms may consider factors such as qubit coherence times, gate fidelity, and the operational capacity of quantum gates to dynamically allocate computational tasks. This adaptive approach ensures that tasks are assigned to quantum processors that are best suited to handle them, maximizing processing speed and efficiency while minimizing the effects of quantum decoherence. The algorithms may be designed to make real-time adjustments based on ongoing performance metrics, allowing for continuous optimization of task distribution in response to fluctuating computational loads and changes in system conditions. By efficiently mapping language processing tasks onto available quantum resources, the Quantum-Classical Integration Module 100 maximizes throughput and minimizes wait times.

The Quantum-Classical Integration Module 100 may also incorporate error mitigation and fault-tolerance techniques to address the inherent noise and uncertainties in quantum systems. The module 100 may utilize quantum error correction codes, such as the surface code or the colour code, to detect and rectify errors that may occur during quantum computations. By integrating these error mitigation strategies, enhancement of the reliability and accuracy of the language processing outcomes is achieved.

To facilitate integration with classical computing infrastructure, the module 100 may incorporate a comprehensive set of APIs and programming interfaces. These interfaces may enable classical systems to invoke quantum algorithms, submit language processing tasks,

and retrieve the results. The module 100 may support programming languages and frameworks utilized in classical natural language processing, such as Python, Java, and TensorFlow, allowing developers to incorporate quantum-enhanced language processing capabilities into their existing workflows.

- 5 In embodiments, the classical data interface 102 is configured to receive data from a classical computing environment and serves as an interface for the ingress of conventional computing data, which includes handling various data formats and ensuring compatibility with existing infrastructure. In embodiments, the data encoding module 104 may be configured to encode received data into a format compatible with a quantum computing environment, incorporating mechanisms to transform raw data into a structured dataset. For example, this transformation includes compliance with metadata standards such as ISO 11179 compliant JSON representation

To ensure scalability and adaptability, the Quantum-Classical Integration Module 100 may be configured to support a wide range of quantum computing platforms and architectures. For example, module 100 can provide a hardware-agnostic abstraction layer that allows the quantum computing framework to interact with different quantum processors, including superconducting qubits, trapped ions, or photonic systems. This flexibility enables the quantum computing framework to leverage the most suitable quantum hardware for each language processing task, utilizing the unique strengths and capabilities of different quantum technologies.

The module 100 may incorporate advanced techniques for data compression and encoding to optimize the transfer of language data between classical and quantum systems. The module 100 may utilize efficient compression algorithms, such as quantum-inspired data compression or quantum source coding, to reduce the size of language data without compromising information content. This compression enables faster data transfer and reduces storage requirements on both classical and quantum hardware. In an embodiment, a large language model may operate on a classical system to generate customer reviews. The Quantum-Classical Integration Module 100 may be configured to improve the sentiment analysis of the customer reviews and enable seamless integration between the classical and quantum systems.

Quantum Processing Module

In example embodiments, the quantum computing framework may utilize a quantum processing module 106 for executing quantum computing via the use of quantum neural networks (QNNs) and tensor networks. This enables data processing in a quantum-

enhanced format. The quantum processing module may use quantum parallelism and superposition principles, enabling the quantum processing module to simultaneously process multiple interpretive pathways. This capability facilitates the extraction of deeper semantic associations, a task that presents significant challenges for classical computing systems. Such an approach utilizes quantum cognitive computing to advance narrative processing capabilities, providing an advancement in the application of quantum technologies to natural language processing. To overcome the constraints of classical computing, the quantum processing module may employ the quantum neural networks to specifically process the narratives generated by a neuromorphic reasoning core, also referred to as a Generative Enhancement through Introspective Governance and Explainable Adaptive Cross-domain Reasoning module or "COGNIGEN-AX," into a quantum-enhanced format. By surpassing the constraints of classical computing, the framework allows COGNIGEN-AX to enhance its cognitive and generative capabilities significantly, facilitating more sophisticated introspection and adaptability across various domains.

In example embodiments, a quantum sentiment analysis function of the quantum processing module 106 may act as an interface between the quantum and classical systems. The function receives customer reviews generated by the language model on the classical system, converts them into JSON representations compliant with ISO 11179 standards, and transfers them to the quantum system. A quantum sentiment analysis circuit within the quantum processing module 106 can be utilized to process the reviews using quantum algorithms. The outcomes may be interpreted, encoded back into JSON format, and sent back to the classical system for additional analysis or integration. The module 100 facilitates the transfer of data between classical and quantum systems, thereby allowing for efficient processing of language model outputs on a large scale and harnessing the computational capabilities of quantum computing for sentiment analysis. By enabling this integration of classical and quantum systems, the module 100 facilitates the framework to leverage the respective advantages of both computing paradigms. This results in improved performance, scalability, and accuracy in natural language processing tasks.

In examples, the task scheduler employed by the module 100 can be configured as a Quantum Advantage Determination Engine (QUADE) which intelligently determines which algorithms better suit quantum versus classical processing for specific language tasks. QUADE may employ several techniques to make this determination:

Complexity Analysis:

QUADE may perform a computational complexity analysis of the target NLP task. This analysis considers various factors such as the size of the input data, the accuracy requirements, and latency constraints. QUADE may then assess the computational complexity of both available quantum and classical algorithms for the task. Quantum algorithms that show potential for polynomial or exponential speedups compared to their classical counterparts are typically favoured for execution on quantum systems. For example, QUADE may conclude that quantum subspace sampling offers an exponential speedup over traditional classical Monte Carlo methods for sentiment analysis tasks involving large datasets.

Quantum Resource Estimation:

QUADE may estimate the necessary quantum resources like qubit count, circuit depth, gate fidelities, and coherence times needed for executing candidate quantum algorithms. QUADE may assess these requirements against the capabilities of existing quantum hardware to determine feasibility. Algorithms that can be reliably executed within the constraints of the available hardware are given priority. For example, QUADE might prefer variational quantum classifiers over fully coherent quantum support vector machines (QSVMs) for text classification tasks if the available quantum processing unit (QPU) only supports limited coherence times.

Quantum Noise Simulation:

QUADE may employ a Quantum Noise Simulation function to simulate the effects of realistic quantum noise models, including amplitude damping, phase flip, and depolarization, on the performance of quantum algorithms. The Quantum Noise Simulation function may assess the noise resilience of various algorithms, prioritizing those that maintain acceptable levels of accuracy and speedup even in noisy quantum conditions. For tasks like named entity recognition (NER), QUADE may favour quantum convolutional models over quantum long short-term memory networks (LSTMs) due to their superior resilience to gate errors.

Classical Intractability Analysis:

QUADE may employ a Classical Intractability Analysis to examine the feasibility of solving NLP tasks with classical algorithms using metrics like sample complexity, hardness of approximation, and classical query lower bounds. Tasks that are found to be classically intractable under realistic conditions are assigned to quantum processing. For instance, when expanding synonyms within large knowledge graphs, QUADE may utilize quantum random walk algorithms to effectively address the classically intractable problem of graph isomorphism.

Hybrid Algorithm Optimization:

QUADE may employ a Hybrid Algorithm Optimization to manage complex NLP pipelines by breaking tasks into subcomponents that are optimally processed on quantum or classical hardware. The Hybrid Algorithm Optimization involves employing heterogeneous computing techniques such as dynamic scheduling, load balancing, and data/model parallelism to enhance the performance of hybrid algorithms end-to-end. For example, in neural machine translation, QUADE could assign the computation of attention mechanisms to quantum processors while processing embedding lookups and beam searches on classical systems, optimizing the utilization of both quantum and classical computational resources.

In example embodiments, Quantum Support Vector Machines (QSVMs) may be implemented by Quantum Processing Module 106 for Natural Language Processing (NLP) tasks such as sentiment analysis. QSVMs are quantum versions of classical support vector machines that provide speedups for text classification tasks like sentiment analysis. In example embodiments, in order to handle text data, pre-trained word embeddings are used to map each word to a quantum state. A sentence may be then encoded as a superposition of its word states. For example, if a sentence has n words $[w_1, w_2, \dots, w_n]$, and each word w_i is mapped to a d -dimensional embedding vector v_i , the sentence can be encoded into a quantum state: $|\psi\rangle = \sum_i \alpha_i |v_i\rangle$. Where α_i are amplitude encoding factors based on the embedding vectors v_i . The QSVM training then aims to find a quantum hyperplane that optimally separates the encoded quantum states of different sentiment classes (e.g., positive and negative). This may be achieved using quantum kernel methods or quantum variational circuits parameterized by the hyperplane coefficients.

In example embodiments, Quantum Neural Networks (QNNs) may be implemented by the Quantum Processing Module 106 for NLP tasks. QNNs are quantum versions of classical neural networks that can provide speed improvements for various NLP tasks, including sentiment analysis. For sentiment analysis, a QNN may take tokenized text (such as word or subword embeddings) as input and encode it into quantum states using amplitude or angle encoding. These encoded quantum states can then be processed by parameterized quantum circuit layers, which act as the hidden layers of the QNN. This approach may use variational quantum circuits, optimizing the circuit parameters using hybrid quantum-classical algorithms like the variational quantum eigensolver (VQE) or quantum approximate optimization algorithm (QAOA). The objective of the optimization is to minimize the error between the QNN's output quantum states and the desired sentiment labels. Techniques such as quantum convolutional neural networks, quantum recurrent neural networks, or quantum attention mechanisms may be incorporated to capture relevant linguistic patterns and long-range dependencies in the text data. Efficient implementation of QNNs for NLP tasks may include: Encoding of text data into quantum states that can be efficiently

processed on NISQ devices, designing parameterized quantum circuit architectures that can effectively model the language task while adhering to hardware constraints, hybrid quantum-classical optimization algorithms to train the QNN parameters and mitigate noise and errors, and techniques like quantum data re-uploading, quantum error correction, and quantum error mitigation to improve overall computation fidelity.

Application Programming Interfaces (APIs) may be integrated within the classical data interface 102 to enable classical systems to interact with quantum computing (QC) algorithms. These APIs may be designed to simplify the complexities of quantum hardware and provide a high-level interface for classical developers to leverage quantum algorithms in their applications. APIs may be structured to support integration between classical systems and QC algorithms in the following manner:

Quantum SDK or Framework: Quantum computing providers may provide a software development kit (SDK) or framework as a foundation for building quantum applications. These SDKs provide a set of libraries, tools, and abstractions to interact with quantum hardware or simulators. Examples of such SDKs include Qiskit (IBM), Cirq (Google), and Q# (Microsoft).

High-Level Programming Languages: APIs may be accessed through high-level programming languages such as Python, C++, or Java. These languages are familiar to classical developers, thereby facilitating the integration of quantum algorithms into their existing codebase.

Quantum Circuit Construction: APIs may provide functions and classes to construct quantum circuits, which are the building blocks of quantum algorithms. Developers can create quantum circuits by specifying quantum gates, measurements, and classical control flow. The API simplifies the complexities associated with the quantum hardware, enabling developers to focus on the algorithmic logic.

Quantum-Classical Hybrid Algorithms: Many quantum algorithms may implement a hybrid approach where classical and quantum computations are interleaved. APIs facilitate this integration by allowing developers to define classical components of an algorithm in their preferred programming language and invoke quantum circuits as needed. The API handles the communication and data transfer between the classical and quantum components.

Job Submission and Result Retrieval: APIs may provide methods to submit quantum circuits or algorithms as jobs to the quantum hardware or simulator. Developers can specify the desired backend (real quantum computer or simulator), the number of shots (repetitions) to

run the circuit, and any additional configuration parameters. The API may process the job submission, queuing, and result retrieval process, abstracting underlying intricacies of the quantum hardware.

- 5 Error Handling and Mitigation: APIs may include error handling mechanisms to deal with the noise and errors present in current quantum hardware. The Error Handling APIs can provide methods for error mitigation techniques, such as readout error mitigation or zero-noise extrapolation, which improve the accuracy of the quantum computations.

- 10 Data Encoding and Decoding: APIs may offer functions to encode classical data into quantum states and decode the measurement results in classical format. This includes encoding data into qubits using amplitude encoding, angle encoding, or quantum feature maps and decoding the measurement outcomes into meaningful classical results.

Integration with Classical Libraries: APIs may be designed to integrate with popular classical libraries and frameworks for data processing, machine learning, and scientific computing. This allows developers to leverage existing tools and libraries with quantum algorithms.

- 15 Error correction may be implemented in noisy intermediate-scale quantum (NISQ) devices within quantum computing module 104. Quantum error correction (QEC) algorithms may be designed to detect and correct errors during quantum computations for achieving reliable results. QEC algorithms may be integrated into NLP tasks in the following manner:

- 20 Quantum Error Correction Codes: Various quantum error correction codes may be developed to detect and correct errors in quantum systems. Some of the codes include the Shor code, Steane code, and surface code. These codes function by encoding logical qubits into multiple physical qubits and using redundancy to detect and correct errors. In NLP tasks, QEC codes may be applied to the quantum circuits that encode text data, perform quantum computations, and measure the results.

- 25 Fault-Tolerant Quantum Computation: Fault-tolerant quantum computation can perform quantum computations reliably in the presence of errors. This computation may include using QEC codes and fault-tolerant protocols to protect the quantum information throughout the calculation. In NLP tasks, fault-tolerant techniques can be employed to ensure the accuracy and reliability of the quantum algorithms used for processing language data.

- 30 Error Mitigation Techniques: In addition to QEC codes, various error mitigation techniques may be implemented to reduce the impact of errors in NISQ devices. These techniques include zero-noise extrapolation, probabilistic error cancellation, and quantum circuit

optimization. Error mitigation can be applied to the quantum circuits used in NLP tasks to improve the accuracy of the results without the need for full-scale QEC.

Quantum Error Correction in Quantum Embeddings: Quantum embeddings, such as quantum word or sentence embeddings, may represent language data in quantum form.

- 5 QEC algorithms may be integrated into creating and manipulating these quantum embeddings. Applying QEC codes to quantum states representing words or sentences may facilitate detection and correction of errors introduced during the embedding process.

Quantum Error Correction in Quantum Neural Networks: Quantum neural networks (QNNs) may be used for NLP tasks such as sentiment analysis and text classification. QEC

- 10 algorithms can be incorporated into the design and training of QNNs to enhance their resilience to errors. Applying QEC codes to the quantum layers and qubits within the QNN can mitigate the impact of errors, leading to more accurate and reliable predictions.

Post-Processing with Error Correction: After performing quantum computations for NLP tasks, the measurement results obtained from the quantum devices may be subject to errors. QEC algorithms can be applied in the post-processing stage to correct these errors. The accuracy of the final results can be improved by using classical error correction techniques in conjunction with the measurement outcomes.

15

FIG. 3 is a flowchart 300 illustrating an example high-level process of using a quantum computing module for processing natural language tasks, according to an embodiment herein.

20

In step 302, input data is encoded into quantum states using a quantum computing module. This encoding translates the structured datasets into a form suitable for quantum operations. The structured datasets may correspond to the datasets that are prepared by a Quantum Data Representation Module in FIG. 4.

25

In step 304, quantum operations are applied via the quantum computing module to the encoded quantum states for processing natural language tasks. These operations may leverage the unique properties of quantum mechanics, such as superposition and entanglement, to perform computations that are infeasible for classical processors.

30

In step 306, the encoded quantum states are measured by quantum circuits of the quantum computing module to determine the results of the natural language processing tasks. This measurement collapses the quantum states into classical states that represent the outcomes of the computations.

In step 308, the results are outputted in a structured dataset suitable for further classical data processing. This allows for easy integration and interpretation within traditional computing environments.

The Quantum Processing Module may implement a collection of quantum algorithms and circuits specifically designed to address computationally intensive tasks related to natural language processing. The Quantum Processing Module may employ unique properties of quantum systems, such as superposition, entanglement, and quantum parallelism, to provide efficiency, accuracy, and scalability in processing and analysing outputs from language models. Example algorithms of the Quantum Processing Module may be configured to provide compatibility with the ISO 11179 compliant JSON representations generated by the Quantum Data Representation Module, enabling processing of a wide range of tasks including semantic analysis and text summarization. The design of the Quantum Processing Module may leverage the capabilities of quantum computing to uncover hidden patterns, identify complex semantic relationships, and extract insights from textual data.

In example embodiments, the Quantum Processing Module may employ a Quantum Latent Semantic Analysis (QLSA) algorithm for semantic analysis. The QLSA may utilize quantum superposition and entanglement to process high-dimensional semantic spaces in an efficient manner. This approach allows the encoding of language model outputs into quantum states, enabling the identification of latent semantic relationships and the generation of concise semantic representations through tailored quantum operations. The quantum-enabled capabilities of QLSA enable the framework to accurately and efficiently capture the underlying meaning and context of textual data. Similarly, a quantum sentiment analysis algorithm utilizes quantum machine learning techniques such as the Quantum Support Vector Machine (QSVM) or Quantum Neural Network (QNN) to classify sentiments expressed in language outputs with precision, providing insights into customer sentiment and brand perception.

In examples, the Quantum Processing Module may employ a Quantum Named Entity Recognition algorithm, which utilizes quantum search and optimization methods to identify and classify named entities (such as individuals, organizations, and geographical locations) within textual data. This capability enhances information retrieval, knowledge discovery, and content organization. For text summarization and content generation purposes, the Quantum Text Summarization and Quantum Language Generation algorithms apply advanced quantum techniques to capture a core essence of texts and generate contextually relevant, semantically rich content. These algorithms can leverage quantum phase estimation,

amplitude amplification, and generative models such as Quantum Generative Adversarial Networks (QGANs) to generate summaries and novel content that maintain linguistic style and coherence. Moreover, the Quantum Processing module may employ advanced techniques from quantum machine learning, optimization, and information theory with the objective of enhancing the performance of natural language processing tasks. The Quantum Processing module may employ algorithms such as the Quantum Approximate Optimization Algorithm (QAOA) and Quantum Annealing to address optimization problems, surpassing classical optimization methods by efficiently exploring extensive search spaces. Quantum information theory principles, like quantum entropy and quantum channel capacity, may be employed to quantify information content and dependencies within language data, thereby assisting in the development of more effective quantum algorithms.

In addition to Quantum Support Vector Machines (QSVMs) and Quantum Generative Adversarial Networks (QGANs), the Quantum Processing Module may employ Quantum Variational Autoencoders (QVAEs) within its suite of quantum operations. Quantum Variational Autoencoders are advanced quantum algorithms that leverage the principles of variational quantum circuits to perform complex encoding and decoding tasks. These encoding and decoding tasks can efficiently compress and decompress large datasets, making QVAEs useful in scenarios involving significant data dimensionality reduction and feature extraction in an inherently quantum fashion. QVAEs may operate by encoding input data into a compressed latent quantum space and subsequently decoding it back to reconstruct the input as closely as possible, thus facilitating data denoising and dimensionality reduction. This process may be enhanced by the quantum nature of the algorithm, which allows it to explore a vast computational space more efficiently than classical autoencoders, providing superior performance in learning compact representations of high-dimensional data.

The Quantum Processing Module's architecture may be designed to be scalable and adaptable, thereby facilitating an integration of emerging quantum algorithms and circuits. Standardized interfaces and data structures may facilitate compatibility across various quantum computing platforms and programming languages. Performance evaluations and benchmarking studies may be conducted to demonstrate the Quantum Processing Module's efficiency and effectiveness. These evaluations compare quantum algorithms against their classical counterparts, evaluating computational complexity, runtime performance, and scalability on extensive language datasets.

The Quantum Processing Module may be configured for various natural language processing tasks. These tasks include:

Semantic Analysis:

Quantum algorithms, such as Quantum Latent Semantic Analysis (QLSA), may extract semantic representations from language model outputs. QLSA may use quantum superposition and entanglement to process high-dimensional semantic spaces efficiently. By encoding language model outputs into quantum states and applying quantum operations, QLSA can identify latent semantic relationships and generate concise semantic representations. This results in a more accurate and efficient semantic analysis compared to classical approaches, especially for large-scale language model outputs. For example, when a substantial corpus of text produced by a language model requires analysis to discern the semantic relationships among various concepts, applying the Quantum Latent Semantic Analysis (QLSA) to these language model outputs, can construct a quantum-enhanced semantic space where related concepts are clustered together. This clustering greatly improves the efficiency of exploring and identifying semantic connections. The utility of this quantum-enhanced approach extends to various applications, including content recommendation systems, information retrieval operations, and the construction of expansive knowledge graphs.

In example embodiments, the JSON data adheres to ISO 11179 standards and encapsulates the results of a sentiment classification task performed by a quantum machine learning model. The JSON structure includes several metadata fields that detail various aspects of the classification process. For instance, metadata about the quantum computing device used, the number of quantum measurements (shots), and the specific device parameters are captured. Other quantum-specific metadata fields or functions may include:

A `additionalMetadata` field offers a comprehensive description of the quantum circuit deployed, detailing the specific gates and measurements utilized. A `measurementProbabilities` field records the probabilities of observing each potential quantum state, while a `measuredQubits` field lists the qubits that were measured during the process. A `sentimentLabel` field indicates the predicted sentiment category derived from the input data. A function `create_sentiment_classification_circuit` is responsible for defining the quantum circuit used for sentiment classification, incorporating gates that are parameterized based on the sentiment features of the input data. A `quantum_sentiment_classifier` function processes the input data by encoding it into quantum states, executing the defined quantum circuit on the specified device, and analysing the measurement results to determine the sentiment labels. A `train_quantum_sentiment_classifier` function manages the training process of the quantum sentiment classifier. It initializes the model parameters and iteratively updates them to improve the classifier's accuracy, continuing this process until the model converges or the

maximum number of iterations is reached. This structured approach in the JSON data ensures that each aspect of the quantum computing process is meticulously documented, facilitating detailed analysis and replication of results.

In an embodiment, customer reviews may be processed using the quantum processing module. For example, a quantum sentiment classifier of the quantum processing module may be initially trained with labelled data. Following this, a new customer review may be analysed using the trained model to determine its sentiment. In this embodiment, the JSON data may follow the ISO 11179 standard and represents the outcome of a text clustering task executed using the Quantum Approximate Optimization Algorithm (QAOA). The JSON data may include various metadata pertaining to the clustering task, such as the device employed, the number of shots taken, and device parameters. Additional quantum-specific metadata fields or functions may include:

An additionalMetadata field containing a quantum circuit description, specifying the applied gates and measurements. A measurementProbabilities field may provide probabilities of measuring each potential state, while the clusters field represents the document clusters obtained, with each cluster containing a list of associated document IDs. A topics field represents the identified topics, including their IDs, keywords, and corresponding distributions. A text_clustering_cost_function may define a cost function used to evaluate the quality of document clusters based on their similarities. A create_qaoa_circuit function may generate the QAOA circuit with the specified parameters. A quantum_text_clustering function implements the text clustering process using QAOA, preprocessing the documents, creating the problem graph, executing the QAOA circuit on the designated device, and interpreting the measurement results to assign documents to clusters. A quantum_topic_modeling function may perform topic modeling using VQE, which preprocesses documents, creates a problem Hamiltonian, defines the VQE ansatz circuit, runs the VQE algorithm on the specified device, and interprets the VQE result to extract the topic distributions. In an example embodiment, a group of documents may be loaded, text clustering is performed using QAOA to determine cluster assignments, and topic modelling may be executed using the VQE to uncover the underlying topics and their distributions.

Scalability Advantages:

Quantum parallelism enables the simultaneous processing of multiple states or configurations in a single quantum operation. This characteristic of quantum computing offers significant scalability benefits when dealing with large-scale language model outputs.

For example, when analysing a massive dataset of language model outputs, such as billions of sentences or paragraph, classical approaches typically require iterating over each data

point sequentially, which can be computationally intensive and time-consuming. By leveraging quantum parallelism, however, it is possible to encode these language model outputs into quantum states and perform quantum operations on them simultaneously. This approach allows for the parallel processing of multiple data points, significantly reducing the computational complexity and enabling faster analysis of large-scale datasets. Moreover, quantum algorithms often exhibit polynomial or even exponential speedups compared to their classical counterparts. As the size of the language model outputs increases, the quantum speedup becomes more pronounced, making quantum computing an increasingly attractive solution for efficiently processing and analyzing massive amounts of text data. The quantum processing module incorporates advanced quantum techniques, such as quantum machine learning and quantum optimization, to further enhance the performance and scalability of natural language processing tasks. Leveraging the capabilities of quantum computing and ISO 11179 compliant JSON representations, the Quantum Processing Module enables efficient processing, analysis, and generation of language model outputs at unprecedented scales.

The Quantum Processing Module may encode text data and semantic features into quantum states using quantum embedding circuits, quantum language models, and other quantum algorithms:

Quantum Embedding Circuits:

Quantum Embedding Circuits may initially process the raw textual data using natural language processing techniques to acquire word embeddings or contextual embeddings from pre-trained language models such as BERT or GPT. These embeddings, which encompass semantic and contextual information, may be high-dimensional vectors that represent words, phrases, or sentences. Parameterized quantum circuits, known as quantum embedding circuits, may then be utilized to map these embeddings onto quantum states. Techniques such as amplitude encoding, basis encoding, and tensor product encoding may be employed in these quantum circuits to load the embedding vectors onto the amplitudes or states of qubits in a quantum-parallelizable manner. For example, in amplitude encoding, the normalized components of the embedding vector may be used to determine the amplitudes of a superposition state across multiple qubits. These quantum embedding circuits may leverage quantum properties such as superposition and entanglement to represent the semantic and contextual information in an enhanced quantum form.

Quantum Language Models:

Extracting Higher-Level Linguistic Structures: Semantic parsing techniques may be utilized to extract comprehensive linguistic structures from the textual data. These structures may include:

Grammar Rules: The fundamental grammatical rules and syntax of the language are

- 5 extracted, capturing the relationships between various linguistic elements such as nouns, verbs, adjectives, and their dependencies.

Knowledge Graphs: Named entities, concepts, and their relationships are identified and organized into knowledge graphs, representing the semantic knowledge within the textual data.

- 10 Relational Schemas: Complex relational structures, such as subject-verb-object triplets or semantic role labelling, are extracted, capturing the intricate dependencies and relationships between entities and events.

Encoding as Tensor Network States:

Subsequently, these higher-level linguistic structures may be encoded as tensor network states, a powerful representation that can capture the intricate dependencies and compositional nature of language.

15

Tensor Representation: Each linguistic element (e.g., word, phrase, concept) may be represented as a tensor, and the relationships between them are encoded as tensor contractions or tensor network diagrams.

20

Compositional Structure: The compositional nature of language may be naturally represented by the tensor network structure, where sub-tensors can be combined and contracted to form larger, more complex tensors representing higher-level linguistic constructs.

Multilinear Operations: Tensor networks may allow for efficient multilinear algebraic

25

operations, such as tensor contractions and decompositions, which can capture complex linguistic transformations and dependencies.

Quantum Language Models: These tensor network states may be the foundation for constructing quantum language models capable of generating valid linguistic outputs and capturing complex dependencies within the language data.

30

Born Machines: Born Machines may be a quantum machine learning model that can represent probabilistic models as quantum circuits. In language modelling, a Born Machine can encode a probabilistic language model as a quantum circuit, where the circuit

parameters correspond to the transition probabilities between linguistic elements (e.g., words, phrases).

Circuit Structure: The quantum circuit structure can be designed to capture the dependencies and relationships encoded in the tensor network states representing the linguistic structures.

Parameter Learning: The circuit parameters, which encode the transition probabilities, can be trained using quantum machine learning techniques, such as variational quantum circuits and hybrid quantum-classical optimization algorithms.

Quantum Semantic Memories:

Semantic relationships, named entities, and knowledge extracted from the text may be stored in specialized quantum data structures inspired by quantum associative memories and quantum random access memories (QRAMs). These data structures may utilize quantum properties such as superposition and entanglement to encode and retrieve semantic information in a manner that is not conforming to the von-Neumann architecture, thereby enabling quantum parallelism. For example, a QRAM could store key-value pairs, where the keys represent classical identifiers for semantic entities, and the values correspond to their respective quantum embeddings. Quantum search algorithms, like Grover's algorithm, may then efficiently retrieve the pertinent quantum embeddings for a given semantic entity.

Quantum Tensor Circuits:

Compositional linguistic structures, such as parse trees and semantic role labels, may be encoded as high-dimensional tensors, capturing the intricate relationships and dependencies within the language data. These tensors may be subsequently mapped onto quantum tensor circuits, which execute the necessary multilinear algebraic operations in superposition by means of quantum gates. For example, a quantum circuit could carry out tensor contractions and decompositions to process the semantic dependencies encoded in a parse tree tensor, harnessing quantum parallelism for enhanced efficiency.

Quantum Error-Correcting Codes:

Given the encoded quantum states representing language data can be highly complex and susceptible to decoherence, techniques from quantum error correction and fault-tolerant quantum computing are employed to protect the quantum information. The logical quantum states that encapsulate linguistic features may be strategically embedded within meticulously constructed quantum error-correcting codes, such as surface codes or topological codes.

These codes may be specifically designed to cater to the unique properties of language data. By incorporating these error-correcting codes, redundancy may be introduced, allowing for the detection and correction of errors that may arise during quantum computations. This facilitates the integrity of the encoded linguistic information.

- 5 The techniques of encoding higher-level linguistic structures as tensor network states and using quantum language models can be utilized for the purpose of mapping financial regulations into data systems. Here are potential implementations of these methodologies:

Extracting Linguistic Structures from Financial Regulations: Financial regulations, such as banking rules, accounting standards, and compliance guidelines, often involve complex

- 10 language with intricate dependencies and relationships. Semantic parsing techniques can be employed to extract the following linguistic structures from these financial regulations:

Grammar Rules: The underlying grammar and syntax rules that govern the structure of the financial regulations can be extracted, capturing the relationships between various linguistic elements like clauses, conditions, and obligations.

- 15 Knowledge Graphs: Named entities related to financial concepts, institutions, instruments, and their relationships can be identified and organized into knowledge graphs, representing the semantic knowledge encoded in the financial regulations.

Relational Schemas: Complex relational structures, such as subject-condition-obligation triplets or semantic role labelling, can be extracted, capturing the dependencies and

- 20 relationships between financial entities, events, and regulatory requirements.

Encoding Linguistic Structures as Tensor Network States: Once the linguistic structures are extracted, they can be encoded as tensor network states, allowing for efficient representation and manipulation of the intricate dependencies and relationships in financial regulations.

- 25 Tensor Representation: Each linguistic element (e.g., clause, condition, obligation) can be expressed as a tensor, and the relationships between them can be encoded as tensor contractions or tensor network diagrams.

Compositional Structure: The compositional nature of financial regulations can be naturally represented by the tensor network structure, where sub-tensors representing individual

- 30 clauses or conditions can be combined and contracted to form larger, more complex tensors representing the overall regulatory framework.

Multilinear Operations: Tensor networks enable efficient multilinear algebraic operations, such as tensor contractions and decompositions, which can capture intricate transformations and dependencies within financial regulations, facilitating their mapping into data systems.

Quantum Language Models for Financial Regulations:

- 5 The tensor network states representing the linguistic structures of financial regulations can serve as a basis for constructing quantum language models that can aid in mapping these regulations into data systems. Born Machines can be used to encode probabilistic models of financial regulations as quantum circuits. The circuit parameters can correspond to the transition probabilities between linguistic elements, capturing the dependencies and
- 10 relationships encoded in the tensor network states. Quantum Reservoir Computing (QRC) can be adapted to process the tensor network states representing financial regulations. The dynamics of the quantum reservoir can be designed to capture the complex dependencies and relationships within the regulations, and the output of the QRC system can represent the mapping of these regulations into data structures or systems.
- 15 The application of quantum language modelling techniques to financial regulations can offer several benefits:

Efficient Representation and Processing: Tensor network states and quantum language models can enable the efficient representation and processing of complex financial regulations. This may be achieved by leveraging quantum parallelism and the ability to

20 manipulate tensors using multilinear operations.

Capturing Complex Dependencies: The compositional nature of tensor networks and the ability of quantum language models to capture intricate dependencies can help accurately represent and map the complex relationships and conditions found in financial regulations.

Interpretability and Explainability: Quantum language models, combined with techniques

25 such as tensor network decompositions, can potentially enhance the interpretability and explainability of financial regulations. This facilitates their mapping into data systems and assists in compliance and risk management processes.

Integration with Quantum Data Systems: As quantum computing technologies progress, the encoded tensor network states and quantum language models can be directly integrated

30 with quantum data systems. This enables the efficient storage, retrieval, and processing of financial regulations within a quantum computing environment.

The Quantum Processing Module may utilize various types of quantum gates and circuits to manipulate and process the quantum states that encode the textual data and semantic

features extracted from large language model outputs. These gates and circuits may be designed to exploit the computational advantages of quantum computing while determining and reasoning about the underlying semantic relationships present in the language data.

Quantum Natural Language Circuits are parameterized quantum circuits that correspond to neural network architectures such as LSTMs, Transformers, etc., but operate in the quantum domain. They may include specialized quantum gates such as quantum convolutions, quantum attention, and quantum recurrent units. These circuits can capture long-range semantic dependencies and relationships by working on the amplitude-encoded quantum states of words and sentences.

Quantum Tensor Network Circuits, which are compositional parse tree structures that encode semantic roles, and relations, are mapped to high-dimensional tensor network states. These circuits may involve gates such as controlled-controlled units perform efficient multilinear algebraic operations on these networks in superposition. This allows semantic compositionality to be captured in a quantum parallelized manner.

Quantum Associative Memory Circuits may encode knowledge of semantic entity-relationship in specialized quantum associative memories and quantum RAM data structures. Circuits involving quantum random access, swap tests, and Toffoli gates implement superposed knowledge retrieval and reasoning over these memories. This efficiently enables the identification of semantic associations such as entity co-occurrences, relational implications, etc.

Once the semantic parsing and relationship extraction is completed using the above circuits, quantum circuits like the Quantum Approximate Optimization Algorithm (QAOA) may be employed. These Quantum Sampling Circuits may sample from the resulting quantum semantic state to efficiently generate linguistic outputs that exhibit valid semantic characteristics and relationships.

Quantum error correction circuits may address the complexities of processing intricate language data by incorporating quantum error correction and fault-tolerant circuits. Utilizing techniques such as Calderbank-Shor-Steane code and topological code, among other quantum error-correcting code (QECC) methods, these circuits protect the semantic information encoded within quantum states from decoherence and operational noise.

The Quantum Processing Module may utilize a series of procedures to map sentiment features extracted from outputs of large language models onto quantum states, enabling quantum-enhanced sentiment analysis. Provided herein is a high-level overview of the process:

Sentiment Feature Extraction:

Classical NLP techniques, including sentiment lexicons, rule-based models, and pre-trained neural networks, may be initially employed to extract sentiment-bearing terms, phrases, and polarity scores from the text. Techniques such as aspect-based sentiment analysis may be further utilized to identify the targets/entities and their associated sentiments.

Sentiment Embedding:

The extracted sentiment features may be projected into high-dimensional embeddings that capture sentiment information and semantic/contextual nuances. Techniques like SentiWordNet, SentiBERT, and other sentiment-specialized embedding models may be adapted for this embedding step.

Quantum Sentiment Embedding Circuits:

The sentiment embeddings may be then mapped to quantum states using specialized parameterized quantum circuits designed specifically for sentiment analysis tasks. These circuits may involve quantum operators, such as IQP (Instantaneous Quantum Polynomial) circuit blocks, which have been theoretically proven to offer advantages in sentiment classification.

Sentiment Feature Encoding:

The multiple sentiment embeddings for a text segment may be jointly encoded using techniques such as: Amplitude encoding into multi-qubit quantum states, Basis encoding as quantum computational basis states, and Tensor product encoding into larger quantum registers.

Sentiment Subspace Mapping:

As the encoded quantum states can be high-dimensional, techniques from quantum principal component analysis (qPCA) and quantum matrix factorizations may be applied. These techniques map the states onto optimal lower-dimensional sentiment subspaces that are maximally expressive while remaining qubit-efficient.

Sentiment Knowledge Enrichment:

External linguistic/domain knowledge resources that capture deeper semantic associations between sentiment terms may be integrated. This involves encoding sentiment knowledge graphs as quantum associative memory structures using techniques like QRAMs.

Quantum Error Correction:

The quantum states used for sentiment encoding may be protected against decoherence using quantum error-correcting codes such as Shor's, Steane's, etc. The choice of quantum error-correcting codes depends on the semantic/sentiment error models that have been characterized for the language data.

- 5 To effectively process large language model outputs, the implementation of quantum algorithms and circuits in the Quantum Data Representation Module may benefit from quantum hardware with specific essential capabilities:

Qubit Count: The Quantum Data Representation Module may encode semantic and sentiment information from language into quantum states, which requires a significant
10 number of qubits, potentially in the thousands or more. Quantum processors with a high qubit count would be necessary, such as those utilizing: Superconducting circuit architectures with large qubit layouts, such as Google Sycamore (53 qubits) and IBM Quantum Condor (1121 qubits), Ion trap architectures that are highly scalable, like IonQ (32 qubits) and Honeywell (32 qubits), and Quantum photonic chips capable of integrating large
15 qubit counts, for example, Xanadu's Borealis with 216 qubits

Qubit Connectivity: Many quantum circuits, including tensor networks and associative memories, require high qubit connectivity to efficiently capture semantic compositionality and relationships. Hardware with high connectivity, such as: Ion trap architectures with all-to-all connectivity, Superconducting chips with rectangular 2D/3D lattice architectures, and
20 Quantum photonic chips with photonic cluster state connectivity.

High-Fidelity Gates: Sentiment analysis often involves deeper and more complex quantum circuits that are susceptible to noise. Hardware with low gate error rates and high coherence times, such as: Ion trap QPUs with ionized atomic clock fidelities (>99.9% for IonQ), Boeing/IBM superconducting architectures (>99.9% 2Q gate fidelities), and Cold atomic
25 QPUs using Rydberg arrays.

Quantum Error Correction: The implementation of efficient quantum error correction can be achieved by encoding complex semantic and linguistic information in quantum states. This may include the use of Quantum Processing Units (QPUs) with high-density qubit arrangements that are capable of supporting Quantum Error Correction (QEC) codes, such
30 as Superconducting Quantum Error Correction Codes (QECCs) that utilize surface/toric codes, or Ion Trap Quantum Error Correction Codes (QECCs) that leverage colour codes or topological cluster state codes.

Quantum Control: Adaptive and controllable quantum circuits may be implemented to capture the dynamic and compositional aspects of language. Architectures that support

quantum control, such as pulse-level analogue controls, may be implemented, including: Superconducting QPUs with microwave/RF quantum control planes, Ion trap QPUs with optical/laser quantum control, and Quantum annealers with programmable parameter control.

- 5 Heterogeneous Computing capabilities may be enabled to orchestrate hybrid quantum-classical workflows, efficiently distributing tasks across the computing spectrum. This includes protocols for sharing memory abstractions like SHMEM and PGAS models, along with maintaining unified intermediate representations for both classical and quantum programs and data. This hybrid setup optimizes computational efficiency and resource
10 utilization.

The module's approach to Scheduling and Resource Management includes co-scheduling of classical data preprocessing and postprocessing with quantum computational tasks. It integrates with established batch schedulers such as Slurm and Kubernetes, which facilitates efficient pooling of computational resources. Additionally, it supports shared job
15 queues and reservation protocols that span classical high-performance computing (HPC) setups and quantum processing units (QPUs).

Computational Storage strategies may be implemented to bring compute capabilities closer to data storage, significantly reducing data movement and latency. This involves offloading tasks like embedding and encoding directly to storage units, integrating computational
20 storage drives, and in-storage processing capabilities through FPGA or ASIC units. Extensions may be made to key-value stores and object stores to include quantum encoding and processing capabilities.

Networking Integration utilizes advanced classical networking technologies including RDMA, DPDK, and SmartNICs, alongside storage/memory fabric technologies such as GenZ, CXL, and OpenCAPI. These integrations are supported by networking middleware like UCX and libfabric, enhancing the module's ability to manage data flow efficiently.
25

The Software Stack Integration may involve the development of APIs, libraries, and compilers that facilitate interoperability between classical and quantum programs. A unified runtime or virtual machine manages both classical and quantum execution, while debuggers
30 and profilers may be extended to support hybrid quantum-classical workflows. This comprehensive software integration ensures that developers can efficiently manage and debug complex integrated systems.

The quantum computing framework may utilize various optimization techniques to determine the optimal parameters for the quantum circuits used in processing and analysing encoded language data:

Quantum Circuit Structure Learning: This involves employing techniques corresponding to neural architecture search and AutoML to automatically learn the optimal structure of quantum circuits (gate layout, qubit topology, etc.) tailored for the characteristics of the language data tensor representations. Specialized search algorithms efficiently navigate the exponentially large quantum circuit search space, guided by complexity measures, entanglement metrics, and computational advantage scores.

Quantum Circuit Parameter Optimization: Once the circuit structure is determined, numerical optimization techniques may be used to optimize the continuous gate parameters of the quantum circuits (rotation angles, phase factors, etc.). Quantum variants of classical optimizers like quantum approximate optimization, quantum evolutionary strategies, and quantum gradient descent may be utilized. Objective functions can include minimizing the length/depth of the quantum circuit, maximizing semantic/linguistic expressivity, and minimizing the impact of decoherence/noise.

Quantum Tensor Network Contractors: For tensor network-based quantum circuits that encode compositional language structures, the module leverages the latest advancements in automated tensor network contraction and optimization algorithms. Techniques such as multi-linear algebra decompositions, contracting tensor networks via tree-based decompositions and DMRG, and co-designing optimized tensor layouts for target quantum hardware topologies may be utilized.

Quantum Control Optimization: In the case of quantum processors with dynamic programmable control planes (e.g., cross-resonant gates in superconductors), the module incorporates quantum optimal control theory to tailor the pulse sequences that control the quantum circuits. Techniques like GRAPE and GOAT may be employed to optimize parameters such as shaped pulses, driving, and tuneable interactions to efficiently and robustly implement target quantum operations.

Codesign of Quantum Circuits and QEC Codes: The selection of quantum error-correcting codes used to protect the encoded language data may be co-optimized along with the quantum circuit parameters. Techniques such as co-design of operation sequences and QEC codes provide an integrated optimization approach to balance factors such as logical qubit overhead, tolerable error rates, and achieving desired fidelities.

Quantum Linguistic Knowledge Distillation: If large pre-trained models that encode linguistic knowledge (e.g., BERT, GPT) are available, the module may employ techniques to distill and compile this knowledge into the parameterized quantum circuit structures. This allows for optimized warm-start initialization of circuit parameters guided by the pre-trained classical linguistic knowledge.

Quantum computing integration with neuromorphic reasoning core:

As quantum computing technologies advance, the neuromorphic reasoning core, COGNIGEN-AX, can be configured to leverage the computational capabilities of quantum processors within the quantum computing module. This integration is designed to enhance the functionality and performance of the quantum computing module:

REIMAS (Retrospective Experience-driven Iterative Multi-model Agent Self-optimization): The REIMAS module enables AI agents to interpret multi-format feedback on their outputs, generate self-reflective critiques, and dynamically adapt their knowledge models and inferential strategies. Adapting REIMAS to utilize quantum algorithms can enhance its ability to process feedback and optimize reasoning processes rapidly, dealing effectively with complex, high-dimensional data spaces.

IPASE (Introspective Performance Analysis and Self-Enhancement): Focused on conducting structured self-critiques and targeted optimization, IPASE can benefit from quantum computing to accelerate the evaluation and optimization cycles, enabling faster and more efficient performance enhancements.

COGNATE (Composable Game-theoretic Nash-embedded Adaptive Techniques for Estimating Confidence): Integrating quantum techniques could refine COGNATE's ability to perform confidence estimation by leveraging quantum-enhanced computation for more accurate and robust decision-making processes.

Additionally, the Petri Net-based orchestration within COGNIGEN-AX will be extended to manage the coordination between quantum and classical computing resources effectively. This capability will enable COGNIGEN-AX to tackle more complex problems and improve performance through quantum-enhanced processing.

Quantum Security and Encryption (QSEM):

As detailed herein, a GenFoundry system, referred to as the "Generative Foundation for Metadata Management, Reporting Alignment, and Data Access Control," is a multi-agent neural architecture that enhances capabilities such as automated metadata management, dynamic data access control, intelligent semantic search, and comprehensive data

governance across various analytical datasets. Within the GenFoundry system, a Quantum Security and Encryption Module (QSEM) serves as an advanced, agent-based component. Moreover, the GenFoundry system, including QSEM, can be integrated into the broader quantum computing framework of the present disclosure.

- 5 Within the quantum computing framework, QSEM utilizes various specialized agents that collaborate within a quantum state orchestration model, executing continuous optimization cycles to ensure robust security measures. The module provides security for encoded data features by integrating quantum encryption algorithms, quantum key distribution (QKD), and post-quantum cryptography. It further employs quantum random number generation and
10 quantum state mapping techniques to enhance the robustness of its encryption capabilities.

To augment its security features, QSEM incorporates quantum-resilient digital signatures to protect data integrity, ensuring the authenticity and non-repudiation of data access and transformations within the GenFoundry system. This approach leverages the unique properties of quantum computing to maintain secure, reliable digital signatures and robust
15 encryption mechanisms, thus enhancing the overall security architecture of the GenFoundry system within the quantum computing framework.

QSEM provides robust data security using quantum encryption algorithms, quantum key distribution, and post-quantum cryptography. QSEM may employs quantum random number generation and quantum state mapping techniques to enhance encryption strength.
20 Additionally, QSEM ensures data integrity through quantum-resilient digital signatures.

QSEM agents:

The QSEM agent architecture may include three key agents, each with specific roles and responsibilities, working together to ensure robust and future-proof data security:

25 QUARK (Quantum Encryption and Key Management): QUARK implements cutting-edge quantum encryption algorithms and leverages quantum mechanics principles for robust encryption. It also incorporates Quantum Key Distribution (QKD) techniques for secure key exchange, detecting and preventing eavesdropping attempts.

PRISM (Post-Quantum Resilience and Integrity Security): PRISM employs post-quantum cryptography algorithms designed to withstand attacks from powerful quantum computers, ensuring long-term data security. It may also utilize quantum-based techniques to generate
30 random and unpredictable numbers for secure cryptographic operations.

QUBITS (Quantum State Mapping and Secure Processing): QUBITS leverage quantum state mapping to securely encode and process data features within the quantum computing

domain, enabling advanced quantum algorithms and computations on encrypted data. QUBITS ensure secure and reliable data operations across the organization.

The QUARK, PRISM, and QUBITS agents may be orchestrated using a quantum state orchestration model, which ensures a coordinated and efficient execution of the security and encryption tasks. The model captures the dependencies and interactions between the agents, ensuring that each task is performed in the appropriate sequence and with the necessary inputs and outputs.

Evaluation and Validation implemented by QSEM:

The selection of ground truth data and evaluation metrics may be tailored to align with the specific functionalities and objectives of each QSEM component:

QUARK: QUARK's performance may be evaluated using simulated quantum computing environments, known cryptanalysis attacks, and industry-standard encryption benchmarks. Encryption strength, key sensitivity, and resistance to known attacks are employed to assess QUARK's effectiveness.

PRISM: PRISM's performance may be assessed using simulated quantum computing attacks, known classical and quantum cryptanalysis techniques, and post-quantum cryptography test suites. Metrics such as algorithm security, computational complexity, and resistance to known attacks are employed to evaluate PRISM's effectiveness in providing long-term data security.

QUBITS: QUBITS's performance may be evaluated using quantum circuit simulations, known quantum state manipulation techniques, and quantum algorithm benchmarks. Metrics such as state fidelity, computation accuracy, and resistance to decoherence are employed to measure QUBITS's effectiveness in securely mapping and processing data features within the quantum computing domain.

By employing these three key agents and evaluation methods, QSEM ensures robust, future-proof, and trustworthy security solutions for the encoded data features within the GenFoundry system, enabling secure and reliable data operations across the organization.

QUARK:

QUARK (Quantum Encryption and Key Management) is an advanced agent that implements quantum encryption algorithms and leverages principles of quantum mechanics for robust encryption. It incorporates Quantum Key Distribution (QKD) techniques for secure key exchange and detects and prevents eavesdropping attempts.

QUARK employs an ensemble of specialized agents that may operate together through continuous optimization cycles, overseen by a quantum encryption orchestration model. The QUARK agents, including QuantumEncryptionAlgorithmSelector, QuantumKeyGenerator, QuantumKeyDistributor, QuantumDecryptionProcessor, and

5 QuantumKeyManagementSystem, operate to implement and manage quantum encryption and key management processes.

QUARK's capabilities for selecting quantum encryption algorithms leverage advanced techniques such as quantum-resistant cryptography, quantum random number generation, and quantum-secure hash functions to identify and implement resilient encryption algorithms
10 against classical and quantum computing attacks. These algorithms may be chosen based on their security properties, performance characteristics, and compatibility with existing systems.

Furthermore, QUARK may incorporate quantum key generation techniques that harness the inherent randomness and unpredictability of quantum systems to generate truly random and
15 secure encryption keys. By leveraging principles of quantum mechanics, such as superposition and entanglement, QUARK ensures that the generated keys are resistant to unauthorized duplication or prediction.

Supporting QUARK's capabilities for quantum key distribution may be its use of QKD protocols, such as BB84 or E91, to securely exchange encryption keys between parties over
20 untrusted channels. By encoding the key information into quantum states and transmitting them over quantum channels, QUARK can detect any attempted eavesdropping or tampering, ensuring the confidentiality and integrity of the distributed keys.

With its quantum decryption processing capabilities, QUARK enables the secure and efficient decryption of encrypted data using quantum-generated keys. It employs optimized
25 quantum circuits and algorithms to perform the decryption operations, harnessing the power of quantum computing to achieve high-speed and high-fidelity decryption results.

QUARK's quantum key management system provides a secure and scalable infrastructure for managing the lifecycle of quantum-generated encryption keys. It includes functionalities for key storage, key revocation, key rotation, and key usage tracking, ensuring the proper
30 handling and protection of the keys throughout their lifespan.

QUARK's architecture may align with the principles of quantum cryptography, quantum information theory, and quantum-safe security. Its modular design and well-defined interfaces enable seamless integration with other components of the Quantum Security and Encryption Module (QSEM) and the broader GenFoundry system.

Processes employed by QUARK

QUARK employs advanced quantum encryption algorithm selection, quantum key generation, quantum key distribution, quantum decryption processing, and quantum key management techniques to ensure robust encryption and secure key exchange in the face of classical and quantum computing threats.

Quantum Encryption Algorithm Selection: QUARK selects and implements quantum encryption algorithms resilient against classical and quantum computing attacks. The quantum encryption algorithm selection process involves the following steps:

Algorithm Analysis: QUARK analyses various quantum encryption algorithms, such as quantum-resistant cryptography, quantum random number generation, and quantum-secure hash functions, to assess their security properties, performance characteristics, and compatibility with existing systems.

Security Evaluation: The QuantumEncryptionAlgorithmSelector agent evaluates the selected algorithms against known quantum attack scenarios, such as Shor's or Grover's algorithms, to ensure their resilience against quantum computing threats.

Performance Benchmarking: QUARK benchmarks the performance of the selected algorithms in terms of key generation time, encryption/decryption speed, and resource consumption to ensure their practical feasibility.

Algorithm Implementation: QUARK implements the selected quantum encryption algorithms using optimized and secure code, following best practices for quantum cryptographic implementations to minimize the risk of side-channel attacks or implementation flaws. The time complexity of the quantum encryption algorithm selection process depends on the number and complexity of the algorithms being analysed, the depth of the security evaluation, and the performance benchmarking procedures. The space complexity may be determined by the size of the quantum cryptographic parameters and the memory requirements of the implemented algorithms.

Quantum Key Generation: QUARK generates truly random and secure encryption keys using quantum-based techniques to ensure the confidentiality and integrity of encrypted data. The quantum key generation process involves the following steps:

Quantum Randomness Source: QUARK employs a quantum randomness source, such as a quantum random number generator or a quantum entanglement-based key generation system, to generate raw random bits based on the inherent randomness and unpredictability of quantum systems.

Quantum Key Encoding: The QuantumKeyGenerator agent encodes the raw random bits into quantum states, such as polarization states of photons or spin states of electrons, using quantum encoding techniques like quantum state preparation or quantum state tomography.

Quantum Error Correction: QUARK applies quantum error correction codes, such as surface codes or topological codes, to the encoded quantum states to protect them against noise and decoherence, ensuring the integrity and reliability of the generated keys.

Key Extraction and Post-Processing: QUARK performs key extraction and post-processing techniques, such as privacy amplification and key reconciliation, to convert the quantum-encoded key states into classical binary key strings suitable for use in encryption algorithms.

The time complexity of the quantum key generation process may depend on the quantum randomness source's generation rate, the efficiency of the quantum encoding and error correction techniques, and the complexity of the key extraction and post-processing procedures. The space complexity may be determined by the size of the quantum key states and the storage requirements for the generated encryption keys.

Quantum Key Distribution: QUARK securely exchanges encryption keys between parties over untrusted channels using quantum key distribution (QKD) protocols. The quantum key distribution process involves the following steps:

QKD Protocol Selection: QUARK selects an appropriate QKD protocol, such as BB84 or E91, based on the available quantum hardware, network topology, and security requirements.

Quantum Channel Establishment: The QuantumKeyDistributor agent establishes a quantum communication channel between the parties involved in the key exchange, using quantum optical fibers, free-space quantum links, or satellite-based quantum networks.

Quantum Key Transmission: QUARK transmits the quantum-encoded key states over the established quantum channel, using the chosen QKD protocol to ensure the security and integrity of the key exchange process.

Eavesdropping Detection and Key Sifting: QUARK leverages the principles of quantum mechanics, such as the no-cloning theorem and the Heisenberg uncertainty principle, to detect any attempted eavesdropping on the quantum channel. It performs key sifting to discard any compromised or mismatched key bits.

Key Confirmation and Authentication: QUARK performs key confirmation and authentication protocols to verify the communicating parties' identity and ensure that the exchanged keys are genuine and have not been tampered with.

The time complexity of the quantum key distribution process may depend on the efficiency of the QKD protocol, the characteristics of the quantum channel, and the complexity of the key sifting and confirmation procedures. The space complexity may be determined by the size of the quantum key states and the storage requirements for the exchanged encryption keys.

- 5 Quantum Decryption Processing: QUARK enables secure and efficient decryption of encrypted data using quantum-generated keys. The Quantum decryption processing involves the following steps:

Quantum Key Retrieval: QUARK retrieves the appropriate quantum-generated encryption key from the quantum key management system based on the encrypted data's associated
10 metadata or key identifier.

Quantum Decryption Circuit Design: The QuantumDecryptionProcessor agent designs an optimized quantum circuit that implements the decryption algorithm corresponding to the encryption scheme used to protect the data.

Quantum Decryption Execution: QUARK executes the designed quantum decryption circuit
15 on a quantum computer or simulator, leveraging the power of quantum parallelism and quantum gate operations to perform the decryption computations efficiently.

Classical Key Extraction: QUARK extracts the decrypted classical key from the quantum decryption circuit's output, using quantum measurement and post-processing techniques to obtain the plaintext key.

- 20 Data Decryption: QUARK uses the decrypted classical key to decrypt the encrypted data using the corresponding classical decryption algorithm, resulting in the recovery of the original plaintext data.

The time complexity of quantum decryption processing may depend on the complexity of the quantum decryption circuit, the efficiency of the quantum hardware or simulator, and the
25 complexity of the classical key extraction and data decryption procedures. The space complexity may be determined by the quantum decryption circuit's size and the decrypted data's storage requirements.

- Quantum Key Management: QUARK provides a secure and scalable infrastructure for managing the lifecycle of quantum-generated encryption keys. The quantum key
30 management process involves the following steps:

Key Storage: To protect the keys from unauthorized access or tampering, the QuantumKeyManagementSystem agent securely stores the quantum-generated encryption

keys using quantum-safe storage techniques, such as quantum key wrapping or quantum secret sharing.

Key Metadata Management: QUARK maintains a metadata repository that associates the stored encryption keys with relevant information, such as key identifiers, key usage policies, expiration dates, and access control permissions.

Key Distribution: QUARK securely distributes the quantum-generated encryption keys to authorized parties using quantum key distribution protocols or quantum-safe communication channels, ensuring the confidentiality and integrity of the key transfer process.

Key Revocation and Rotation: QUARK implements key revocation mechanisms to invalidate compromised or expired keys and performs key rotation procedures to periodically update the encryption keys, enhancing the system's long-term security.

Key Usage Tracking: QUARK monitors and logs the usage of the quantum-generated encryption keys, recording information such as key access events, key usage purposes, and key lifecycle status changes for auditing and compliance purposes.

The quantum key management process's time complexity may depend on the efficiency of the key storage and retrieval operations, the complexity of the key distribution and revocation protocols, and the volume of key usage events being tracked. The space complexity may be determined by the size of the key metadata repository and the storage requirements for the quantum-generated encryption keys.

QUARK Design - Business Layer

Processes:

Quantum Encryption Algorithm Integration: This process involves selecting, implementing, and integrating quantum encryption algorithms resilient against classical and quantum computing attacks to ensure robust encryption of sensitive data.

Quantum Key Generation and Distribution: This process focuses on generating truly random and secure encryption keys using quantum-based techniques and securely distributing them between parties using quantum key distribution (QKD) protocols.

Quantum Decryption and Data Access: This process involves securely and efficiently decrypting encrypted data using quantum-generated keys and providing authorized access to the decrypted plaintext data.

Quantum Key Management and Lifecycle: This process focuses on managing the lifecycle of quantum-generated encryption keys, including key storage, distribution, revocation, rotation, and usage tracking, to ensure the encryption system's long-term security and scalability.

Functions:

- 5 Robust Quantum Encryption: QUARK ensures the robust encryption of sensitive data by employing quantum encryption algorithms resilient against classical and quantum computing attacks, providing long-term data confidentiality.

- Secure Quantum Key Exchange: QUARK enables the secure exchange of encryption keys between parties over untrusted channels using quantum key distribution (QKD) protocols,
10 ensuring the confidentiality and integrity of the distributed keys.

Efficient Quantum Decryption: QUARK enables efficient decryption of encrypted data using quantum-generated keys and optimized quantum circuits, leveraging the power of quantum computing to achieve high-speed and high-fidelity decryption results.

- Scalable Quantum Key Management: QUARK provides a scalable and secure infrastructure
15 for managing the lifecycle of quantum-generated encryption keys, ensuring proper key storage, distribution, revocation, rotation, and usage tracking.

QUARK Design - Application Layer

The QUARK system comprises the following components:

- QuantumEncryptionAlgorithmSelector: This component selects and implements quantum
20 encryption algorithms that are resilient against classical and quantum computing attacks, ensuring robust encryption of sensitive data.

QuantumKeyGenerator: This component generates truly random and secure encryption keys using quantum-based techniques, such as quantum random number generation and quantum state encoding.

- 25 QuantumKeyDistributor: This component securely distributes the quantum-generated encryption keys between parties using quantum key distribution (QKD) protocols, ensuring the confidentiality and integrity of the critical exchange process.

- QuantumDecryptionProcessor: This component enables secure and efficient decryption of encrypted data using quantum-generated keys and optimized quantum circuits, leveraging
30 the power of quantum computing for high-speed decryption.

QuantumKeyManagementSystem: This component provides a secure and scalable infrastructure for managing the lifecycle of quantum-generated encryption keys, including crucial storage, distribution, revocation, rotation, and usage tracking.

Services:

- 5 QuantumEncryptionAlgorithmSelectionService: Provides methods for selecting and implementing quantum encryption algorithms resilient against classical and quantum computing attacks.

- 10 QuantumKeyGenerationService: Offers services for generating genuinely random and secure encryption keys using quantum-based techniques, such as quantum random number generation and quantum state encoding.

QuantumKeyDistributionService: Enables secure distribution of quantum-generated encryption keys between parties using quantum key distribution (QKD) protocols.

QuantumDecryptionProcessingService: Facilitates secure and efficient decryption of encrypted data using quantum-generated keys and optimized quantum circuits.

- 15 QuantumKeyManagementService: Provides services for managing the lifecycle of quantum-generated encryption keys, including crucial storage, distribution, revocation, rotation, and usage tracking.

Interfaces:

- 20 QuantumEncryptionAlgorithmSelectorInterface: This interface defines the methods and parameters for selecting and implementing quantum encryption algorithms that are resilient against both classical and quantum computing attacks.

QuantumKeyGeneratorInterface: Specifies the methods and output formats for generating genuinely random and secure encryption keys using quantum-based techniques.

- 25 QuantumKeyDistributorInterface: Describes the methods and parameters for securely distributing quantum-generated encryption keys between parties using quantum key distribution (QKD) protocols.

QuantumDecryptionProcessorInterface: Defines the methods and input/output formats for securely and efficiently decrypting encrypted data using quantum-generated keys and optimized quantum circuits.

- 30 QuantumKeyManagementSystemInterface: Specifies the methods and parameters for managing the lifecycle of quantum-generated encryption keys, including crucial storage, distribution, revocation, rotation, and usage tracking

PRISM

PRISM (Post-Quantum Resilience and Integrity Security) is an advanced agent that employs post-quantum cryptography algorithms designed to withstand attacks from powerful quantum computers. It ensures long-term data security by utilizing quantum-based techniques to generate random and unpredictable numbers for secure cryptographic operations.

In example embodiments, PRISM employs a synergistic ensemble of specialized agents collaborating through continuous optimization cycles, overseen by a post-quantum security orchestration model. The PRISM agents, including PostQuantumCryptoSelector, QuantumRandomNumberGenerator, QuantumKeyDistributor, IntegrityVerifier, and CryptographicProtocolAnalyzer, work in concert to select, implement, and analyze post-quantum cryptographic algorithms and protocols.

PRISM's post-quantum cryptography selection capabilities leverage advanced techniques like lattice-based cryptography, code-based cryptography, multivariate cryptography, or hash-based signatures to identify and implement cryptographic algorithms resistant to quantum attacks. These algorithms are carefully chosen based on their security properties, performance characteristics, and compatibility with existing systems.

Furthermore, PRISM incorporates quantum random number generation techniques that exploit the inherent randomness of quantum systems, such as quantum shot noise or quantum vacuum fluctuations, to generate truly random and unpredictable numbers. These quantum-generated random numbers serve as the foundation for secure cryptographic key generation, ensuring the strength and integrity of the cryptographic operations.

Underpinning PRISM's quantum key distribution capabilities is its use of quantum communication protocols, such as BB84 or E91, to securely exchange cryptographic keys between parties over untrusted channels. By leveraging the principles of quantum mechanics, such as the no-cloning theorem and the Heisenberg uncertainty principle, PRISM ensures the confidentiality and integrity of the distributed keys, detecting any eavesdropping attempts.

With its cryptographic protocol analysis and verification techniques, PRISM ensures the security and correctness of the post-quantum cryptographic protocols implemented within the system. It employs formal verification methods, such as symbolic or computational analysis, to prove the security protocols' security properties against various attack scenarios and identify vulnerabilities or weaknesses.

PRISM's architecture aligns with the principles of post-quantum cryptography, quantum-enhanced security, and formal verification. Its modular design and well-defined interfaces

enable seamless integration with the other components of the Quantum Security and Encryption Module (QSEM) and the broader GenFoundry system.

Processes employed by PRISM:

PRISM may employ advanced post-quantum cryptography selection, quantum random
5 number generation, quantum key distribution, and cryptographic protocol analysis techniques to ensure long-term data security and integrity in the face of quantum computing threats.

Post-Quantum Cryptography Selection: PRISM selects and implements post-quantum cryptographic algorithms resistant to attacks from powerful quantum computers. The post-
10 quantum cryptography selection process involves the following steps:

Algorithm Analysis: PRISM analyses various post-quantum cryptographic algorithms, such as lattice-based, code-based, multivariate, or hash-based schemes, to assess their security properties, performance characteristics, and compatibility with existing systems.

Security Evaluation: The PostQuantumCryptoSelector agent evaluates the selected
15 algorithms against known quantum attack scenarios, such as Shor's or Grover's algorithms, to ensure their resilience against quantum computing threats.

Performance Benchmarking: PRISM benchmarks the performance of the selected algorithms in terms of critical generation time, encryption/decryption speed, signature generation/verification time, and resource consumption to ensure their practical feasibility.

20 Algorithm Implementation: PRISM implements the selected post-quantum cryptographic algorithms using optimized and secure code, following best practices for cryptographic implementations to minimize the risk of side-channel attacks or implementation flaws.

The time complexity of the post-quantum cryptography selection process may depend on the number and complexity of the algorithms being analysed, the depth of the security
25 evaluation, and the performance benchmarking procedures. The space complexity may be determined by the size of the cryptographic parameters and the memory requirements of the implemented algorithms.

Quantum Random Number Generation: PRISM generates genuinely random and unpredictable numbers using quantum-based techniques to ensure the security of
30 cryptographic operations. The quantum random number generation process involves the following steps:

Quantum Entropy Source: PRISM employs a quantum entropy source, such as a quantum optics setup or a quantum circuit, to generate raw random bits based on the inherent randomness of quantum systems, such as quantum shot noise or quantum vacuum fluctuations.

- 5 Randomness Extraction: The QuantumRandomNumberGenerator agent applies randomness extraction techniques, such as hashing or seeded extractors, to convert the raw quantum random bits into uniform and unbiased random numbers suitable for cryptographic use.

- 10 Statistical Testing: PRISM performs statistical tests, such as the NIST randomness test suite or the dieharder test suite, to assess the quality and unpredictability of the generated random numbers and ensure their suitability for cryptographic applications.

- Secure Storage and Distribution: The generated quantum random numbers are securely stored and distributed to the relevant cryptographic components within the system, using secure communication channels and access control mechanisms to prevent unauthorized
15 access or tampering.

- The time complexity of the quantum random number generation process may depend on the quantum entropy source's generation rate, the efficiency of the randomness extraction techniques, and the complexity of the statistical testing procedures. The space complexity may be determined by the size of the random number buffers and the storage requirements
20 for the generated random numbers.

Quantum Key Distribution: PRISM securely exchanges cryptographic keys between parties over untrusted channels using quantum communication protocols. The quantum critical distribution process involves the following steps:

- Quantum Channel Establishment: PRISM establishes a quantum communication channel
25 between the parties involved in the key exchange, using quantum optical fibers, free-space quantum communication links, or satellite-based quantum networks.

Quantum Key Exchange Protocol: The QuantumKeyDistributor agent implements a quantum key exchange protocol, such as BB84 or E91, to securely transmit quantum states encoding the cryptographic key bits between the parties.

- 30 Eavesdropping Detection: PRISM leverages the principles of quantum mechanics, such as the no-cloning theorem and the Heisenberg uncertainty principle, to detect any eavesdropping attempts on the quantum channel, ensuring the confidentiality and integrity of the exchanged keys.

Critical Reconciliation and Privacy Amplification: PRISM performs essential reconciliation and privacy amplification techniques to correct errors in the exchanged key bits and remove any potential information leakage to eavesdroppers, resulting in a secure and secret shared key.

- 5 The time complexity of the quantum key distribution process may depend on the efficiency of the quantum key exchange protocol, the complexity of the error correction and privacy amplification techniques, and the characteristics of the quantum channel. The space complexity may be determined by the size of the quantum key buffers and the storage requirements for the shared secret keys.

- 10 Cryptographic Protocol Analysis: PRISM ensures the security and correctness of the post-quantum cryptographic protocols implemented within the system through formal verification and analysis techniques. The cryptographic protocol analysis process involves the following steps:

- Protocol Specification: PRISM specifies the post-quantum cryptographic protocols using
15 formal languages, such as applied pi calculus or protocol composition logic, to unambiguously describe the protocol's steps, messages, and cryptographic operations.

- Formal Verification: The CryptographicProtocolAnalyzer agent employs formal verification methods, such as symbolic or computational analysis, to prove the security properties of the specified protocols against various attack scenarios and identify potential vulnerabilities or
20 weaknesses.

Security Property Checking: PRISM verifies the protocols' desired security properties, such as confidentiality, integrity, authentication, or non-repudiation, using automated theorem provers or model checkers to ensure their adherence to their intended security goals.

- Attack Simulation: PRISM simulates various attack scenarios, such as man-in-the-middle
25 attacks, replay attacks, or quantum computing attacks, against the protocols to assess their resilience and identify potential attack vectors that may compromise the system's security.

- The time complexity of the cryptographic protocol analysis process may depend on the complexity of the protocol specifications, the efficiency of the formal verification methods, and the number of security properties and attack scenarios being checked. The space
30 complexity may be determined by the size of the protocol models and the memory requirements of the verification tools and attack simulation engines.

PRISM Design – Business Layer

Processes:

Post-Quantum Cryptography Integration: This process involves selecting, implementing, and integrating post-quantum cryptographic algorithms into the system to ensure long-term data security against quantum computing threats.

5 Quantum Random Number Generation: This process focuses on generating genuinely random and unpredictable numbers using quantum-based techniques to ensure the security of cryptographic operations.

Quantum Key Distribution: This process involves securely exchanging cryptographic keys between parties over untrusted channels using quantum communication protocols, ensuring the confidentiality and integrity of the distributed keys.

10 Cryptographic Protocol Verification: This process formally verifies the security and correctness of the post-quantum cryptographic protocols implemented within the system, identifying potential vulnerabilities or weaknesses.

Functions:

15 Quantum-Resistant Data Security: PRISM ensures the long-term security of sensitive data by employing post-quantum cryptographic algorithms resistant to attacks from powerful quantum computers.

Enhanced Cryptographic Randomness: PRISM enhances the security of cryptographic operations by generating genuinely random and unpredictable numbers using quantum-based techniques, providing a solid foundation for cryptographic key generation.

20 Secure Key Exchange: PRISM enables the secure exchange of cryptographic keys between parties over untrusted channels using quantum communication protocols, ensuring the confidentiality and integrity of the distributed keys.

25 Verified Cryptographic Protocols: PRISM ensures the security and correctness of the post-quantum cryptographic protocols through formal verification and analysis, identifying and mitigating potential vulnerabilities or weaknesses.

PRISM Design – Application Layer

The PRISM system comprises the following components:

30 PostQuantumCryptoSelector: This component selects and implements post-quantum cryptographic algorithms resistant to attacks from powerful quantum computers, ensuring the long-term security of sensitive data.

QuantumRandomNumberGenerator: This component generates genuinely random and unpredictable numbers using quantum-based techniques, providing a solid foundation for secure cryptographic key generation.

5 QuantumKeyDistributor: This component securely exchanges cryptographic keys between parties over untrusted channels using quantum communication protocols, ensuring the confidentiality and integrity of the distributed keys.

IntegrityVerifier: This component verifies the integrity of the exchanged cryptographic keys and the transmitted data using post-quantum cryptographic algorithms and protocols.

10 CryptographicProtocolAnalyzer: This component formally verifies the security and correctness of the post-quantum cryptographic protocols implemented within the system, identifying any potential vulnerabilities or weaknesses.

Services:

15 PostQuantumCryptoSelectionService: Provides methods for selecting and implementing post-quantum cryptographic algorithms resistant to attacks from powerful quantum computers.

QuantumRandomNumberGenerationService: This service offers services for generating truly random and unpredictable numbers using quantum-based techniques, ensuring the security of cryptographic operations.

20 QuantumKeyDistributionService: Enables secure exchange of cryptographic keys between parties over untrusted channels using quantum communication protocols.

IntegrityVerificationService: Facilitates the verification of the integrity of the exchanged cryptographic keys and the transmitted data using post-quantum cryptographic algorithms and protocols.

25 CryptographicProtocolAnalysisService: Provides services for formally verifying the security and correctness of the post-quantum cryptographic protocols implemented within the system.

Interfaces:

PostQuantumCryptoSelectorInterface: Defines the methods and parameters for selecting and implementing post-quantum cryptographic algorithms resistant to quantum attacks.

30 QuantumRandomNumberGeneratorInterface: Specifies the methods and output formats for generating truly random and unpredictable numbers using quantum-based techniques.

QuantumKeyDistributorInterface: This interface describes the methods and parameters for securely exchanging cryptographic keys between parties over untrusted channels using quantum communication protocols.

IntegrityVerifierInterface: Defines the methods and input/output formats for verifying the integrity of the exchanged cryptographic keys and the transmitted data using post-quantum cryptographic algorithms and protocols.

CryptographicProtocolAnalyzerInterface: This interface specifies the methods and input/output formats for formally verifying the security and correctness of the post-quantum cryptographic protocols implemented within the system.

10 QUBITS

QUBITS (Quantum State Mapping and Secure Processing) is an advanced agent that leverages quantum state mapping to securely encode and process data features within the quantum computing domain. It enables advanced quantum algorithms and computations on encrypted data, ensuring secure and reliable data operations across the organization.

15 In implementations, QUBITS employ a synergistic ensemble of specialized agents collaborating through continuous optimization cycles, overseen by a quantum state orchestration model. The QUBITS agents, including QuantumEncoder, SecureQuantumProcessor, QuantumAlgorithmOptimizer, QuantumErrorCorrector, and QuantumResultInterpreter, work in concert to map, process, and secure data features using
20 quantum computing principles.

QUBITS's quantum state mapping capabilities leverage advanced techniques like quantum amplitude encoding, quantum phase encoding, or quantum kernel embedding to represent classical data features as quantum states. This mapping efficiently processes high-dimensional data within the quantum computing domain, enabling complex computations
25 and pattern recognition tasks.

Furthermore, QUBITS incorporate secure quantum processing mechanisms that leverage the inherent properties of quantum systems, such as quantum superposition, entanglement, and quantum key distribution, to perform computations on encrypted data without revealing the underlying information. This ensures the confidentiality and integrity of sensitive data
30 features throughout the quantum processing lifecycle.

QUBITS's quantum algorithm optimization capabilities may be based on its use of variational quantum algorithms, quantum circuit learning, and quantum-classical hybrid approaches. QUBITS optimize the quantum circuits and algorithms used for processing the mapped data

features, considering the specific characteristics of the quantum hardware and the desired computational tasks.

With its quantum error correction and fault-tolerant computing techniques, QUBITS ensure the reliability and robustness of the quantum computations performed on the mapped data features. It employs quantum error correction codes, such as surface or color codes, to detect and correct errors introduced by noise or decoherence in the quantum system.

QUBITS's architecture aligns with the principles of quantum-enhanced security, quantum-accelerated machine learning, and quantum-classical hybrid computing. It's modular design and well-defined interfaces enable seamless integration with the other components of the Quantum Security and Encryption Module (QSEM) and the broader GenFoundry system.

Processes employed by QUBITS:

QUBITS employ advanced quantum state mapping, secure quantum processing, quantum algorithm optimization, and quantum error correction techniques to encode, process, and secure data features within the quantum computing domain.

Quantum State Mapping: QUBITS maps classical data features into quantum states, enabling efficient processing and computation within the quantum domain. The quantum state mapping process involves the following steps:

Data Preprocessing: QUBITS pre-processes the classical data features, such as normalization, feature scaling, or dimensionality reduction, to prepare them for quantum encoding.

Quantum Encoding: The QuantumEncoder agent applies quantum encoding techniques, such as quantum amplitude encoding, quantum phase encoding, or quantum kernel embedding, to map the preprocessed data features into quantum states. This encoding represents the data features in a high-dimensional Hilbert space, enabling quantum parallelism and entanglement.

State Preparation: QUBITS prepare the encoded quantum states on the quantum hardware using state preparation circuits, such as quantum gate sequences or variational state preparation algorithms.

Quantum Circuit Optimization: QUBITS optimize the state preparation circuits to minimize the circuit depth, gate count, and coherence time requirements, taking into account the specific characteristics of the quantum hardware.

The time complexity of the quantum state mapping process may depend on the dimensionality of the data features, the chosen encoding technique, and the efficiency of the state preparation circuits. The number of qubits required to represent the encoded quantum states determines the space complexity.

- 5 Secure Quantum Processing: QUBITS performs secure computations on the mapped data features using quantum algorithms and cryptographic techniques. The secure quantum processing involves the following steps:

Quantum Algorithm Selection: QUBITS select the appropriate quantum algorithms for the desired computational tasks, such as quantum machine learning, quantum optimization, or
10 quantum linear algebra, based on the nature of the data features and the security requirements.

Quantum Circuit Design: The SecureQuantumProcessor agent designs the quantum circuits that implement the selected quantum algorithms, incorporating secure computation techniques like blind quantum computing, homomorphic encryption, or quantum key
15 distribution.

Quantum Circuit Execution: QUBITS execute the designed quantum circuits on the quantum hardware, leveraging quantum parallelism and entanglement to perform the computations efficiently.

Quantum Measurement: The quantum states resulting from the computations are measured
20 using appropriate measurement bases and readout techniques, extracting the relevant classical information while maintaining the security and confidentiality of the data features.

The time complexity of the secure quantum processing may depend on the complexity of the selected quantum algorithms, the size of the quantum circuits, and the efficiency of the quantum hardware. The space complexity may be determined by the number of qubits and
25 the depth of the quantum circuits required for the computations.

Quantum Algorithm Optimization: QUBITS optimize the quantum algorithms and circuits for processing the mapped data features to achieve optimal performance and resource utilization. The quantum algorithm optimization process involves the following steps:

Algorithm Analysis: QUBITS analyzes the selected quantum algorithms to identify
30 opportunities for optimization, such as reducing the circuit depth, minimizing the number of gates, or exploiting the problem's structure.

Variational Optimization: The QuantumAlgorithmOptimizer agent applies variational quantum algorithms, such as variational quantum eigensolvers (VQE) or quantum approximate

optimization algorithms (QAOA), to optimize the parameters of the quantum circuits iteratively.

Quantum Circuit Learning: QUBITS employs quantum circuit learning techniques, such as parameterized quantum circuits or quantum neural networks, to learn the optimal circuit structures and parameters from the data features and the desired computational tasks.

Quantum-Classical Hybrid Approaches: QUBITS leverage quantum-classical hybrid approaches, such as variational quantum classifiers or quantum-assisted optimization, to combine the strengths of quantum and classical computing, enabling efficient and scalable optimization of the quantum algorithms.

The time complexity of the quantum algorithm optimization process may depend on the size and complexity of the quantum circuits, the number of optimization parameters, and the convergence rate of the variational algorithms. The number of qubits and the depth of the quantum circuits involved in the optimization process determine the space complexity.

Quantum Error Correction: QUBITS ensure the reliability and robustness of the quantum computations by employing quantum error correction and fault-tolerant techniques. The quantum error correction process involves the following steps:

Error Syndrome Measurement: The QuantumErrorCorrector agent performs error syndrome measurements on the quantum states to detect the presence and type of errors introduced by noise or decoherence.

Error Correction: Based on the error syndrome measurements, QUBITS apply appropriate error correction procedures, such as quantum error correction codes (e.g., surface codes, color codes) or stabilizer measurements, to correct the detected errors.

Fault-Tolerant Computation: QUBITS employ fault-tolerant quantum computation techniques, such as magic state distillation or topological quantum error correction, to enable reliable computations even in the presence of errors and imperfections in the quantum hardware.

Error Threshold Analysis: QUBITS analyzes the error rates and thresholds of the quantum hardware to determine the feasibility and scalability of the quantum error correction schemes and to optimize the error correction parameters accordingly.

The time complexity of the quantum error correction process may depend on the chosen error correction codes, the error rates of the quantum hardware, and the complexity of the fault-tolerant computation techniques. The space complexity may be determined by the number of qubits required for encoding the logical qubits and the ancillary qubits needed for error syndrome measurements.

QUBITS design – Business Layer

Processes:

Data Feature Mapping: This process involves preprocessing the classical data features and mapping them into quantum states using quantum encoding techniques, enabling efficient processing and computation within the quantum domain.

Secure Quantum Computation: This process focuses on performing secure computations on the mapped data features using quantum algorithms and cryptographic techniques, ensuring the confidentiality and integrity of the data throughout the quantum processing lifecycle.

Quantum Algorithm Optimization: This process involves optimizing the quantum algorithms and circuits used for processing the mapped data features, leveraging variational quantum algorithms, quantum circuit learning, and quantum-classical hybrid approaches to achieve optimal performance and resource utilization.

Quantum Error Correction and Fault Tolerance: This process ensures the reliability and robustness of the quantum computations by employing quantum error correction codes, fault-tolerant computation techniques, and error threshold analysis.

Functions:

Efficient Quantum Data Encoding: QUBITS efficiently encodes classical data features into quantum states, leveraging quantum state mapping techniques to represent high-dimensional data within the quantum computing domain.

Secure Quantum Data Processing: QUBITS ensures the secure processing of mapped data features using quantum algorithms and cryptographic techniques, preserving the confidentiality and integrity of sensitive information throughout the computation lifecycle.

Optimized Quantum Algorithm Execution: QUBITS optimize the quantum algorithms and circuits for processing the mapped data features, achieving optimal performance and resource utilization through variational optimization, quantum circuit learning, and quantum-classical hybrid approaches.

Reliable and Fault-Tolerant Quantum Computation: QUBITS ensure the reliability and robustness of the quantum computations by employing quantum error correction codes and fault-tolerant computation techniques, enabling accurate and trustworthy results even in the presence of noise and imperfections in the quantum hardware.

QUBITS design – Application Layer

The QUBITS system comprises of the following components:

QuantumEncoder: This component applies quantum encoding techniques, such as quantum amplitude encoding, phase encoding, or quantum kernel embedding, to map classical data features into quantum states.

- 5 SecureQuantumProcessor: This component designs and executes quantum circuits that implement secure quantum algorithms, incorporating cryptographic techniques to ensure the confidentiality and integrity of the data during processing.

- QuantumAlgorithmOptimizer: This component optimizes the quantum algorithms and circuits used for processing the mapped data features, leveraging variational quantum algorithms,
10 quantum circuit learning, and quantum-classical hybrid approaches.

QuantumErrorCorrector: This component performs error syndrome measurements, applies quantum error correction codes, and employs fault-tolerant computation techniques to ensure the reliability and robustness of the quantum computations.

- QuantumResultInterpreter: This component interprets the quantum measurement results,
15 extracting the relevant classical information while maintaining the security and confidentiality of the processed data features.

Services:

QuantumStateMappingService: Provides methods for preprocessing classical data features and mapping them into quantum states using quantum encoding techniques.

- 20 SecureQuantumProcessingService: Offers services for designing and executing quantum circuits that implement secure quantum algorithms, ensuring the confidentiality and integrity of the data during processing.

- QuantumAlgorithmOptimizationService: This service enables the optimization of quantum algorithms and circuits for processing the mapped data features, leveraging variational
25 quantum algorithms, quantum circuit learning, and quantum-classical hybrid approaches.

QuantumErrorCorrectionService: Facilitates the application of quantum error correction codes, fault-tolerant computation techniques, and error threshold analysis to ensure the reliability and robustness of the quantum computations.

- QuantumResultInterpretationService: Provides services for interpreting the quantum
30 measurement results and extracting the relevant classical information while maintaining the security and confidentiality of the processed data features.

Interfaces:

QuantumEncoderInterface: Defines the methods and parameters for preprocessing classical data features and mapping them into quantum states using quantum encoding techniques.

SecureQuantumProcessorInterface: Specifies the methods and input/output formats for designing and executing quantum circuits that implement secure quantum algorithms, ensuring the confidentiality and integrity of the data during processing.

QuantumAlgorithmOptimizerInterface: Describes the methods and parameters for optimizing quantum algorithms and circuits to process the mapped data features, leveraging variational quantum algorithms, quantum circuit learning, and quantum-classical hybrid approaches.

QuantumErrorCorrectorInterface: Defines the methods and input/output formats for applying quantum error correction codes, fault-tolerant computation techniques, and error threshold analysis to ensure the reliability and robustness of the quantum computations.

QuantumResultInterpreterInterface: This interface specifies the methods and output formats for interpreting the quantum measurement results and extracting the relevant classical information while maintaining the security and confidentiality of the processed data features.

Cryptographic Audit Trail Module (CATM)

As described herein, a Cryptographic Audit Trail Module (CATM) is an advanced, agent-based system that ensures data integrity, provenance tracking, and transparency within a multi-agent neural architecture such as the GenFoundry system. In examples, the CATM employs a synergistic ensemble of three specialized agents collaborating through continuous optimization cycles, overseen by a distributed ledger orchestration model. CATM generates immutable audit trails that cryptographically capture the provenance of data elements throughout their lifecycle, from originating sources to analytical consumption. It leverages quantum computing techniques for efficient provenance verification at scale and integrates with distributed ledger technologies for transparent, multi-party governance of audit trails.

This multi-agent architecture includes Quantum Verification and Security Assurance (QUASAR) which leverages quantum computing techniques, such as quantum parallelism and quantum algorithms, to optimize provenance verification at scale. It also employs quantum-resilient cryptographic algorithms and post-quantum digital signatures to ensure audit trails' long-term security and integrity. QUASAR's performance may be assessed using simulated quantum computing environments, known quantum optimization techniques, and quantum algorithm benchmarks. Metrics such as verification speedup, scaling efficiency, and resistance to known quantum computing vulnerabilities are employed to evaluate QUASAR's effectiveness in optimizing provenance verification at scale.

QUASAR

QUASAR (Quantum Verification and Security Assurance) is an advanced agent that leverages quantum computing techniques, such as quantum parallelism and quantum algorithms, to optimize provenance verification at scale. It also employs quantum-resilient cryptographic algorithms and post-quantum digital signatures to ensure audit trails' long-term security and integrity.

In examples, QUASAR employs a synergistic ensemble of specialized agents collaborating through continuous optimization cycles overseen by a quantum-enhanced orchestration model. The QUASAR agents, including QuantumVerifier, PostQuantumSecurer, QuantumOptimizer, CryptographicAnalyzer, and IntegrityEnforcer, work in concert to enhance the security, scalability, and efficiency of audit trail verification and protection.

QUASAR's quantum verification capabilities leverage the power of quantum computing to perform parallel verification of audit trail integrity and consistency. By encoding the audit trail data into quantum states and employing quantum algorithms, QUASAR can simultaneously verify multiple aspects of the audit trails, such as cryptographic signatures, data lineage, and temporal consistency, significantly reducing the verification time compared to classical approaches.

Furthermore, QUASAR incorporates post-quantum cryptographic algorithms and digital signatures to protect the audit trails against potential threats posed by quantum computers. These quantum-resilient techniques ensure that the integrity and non-repudiation properties of the audit trails remain secure, even in the presence of adversaries with access to powerful quantum computing capabilities.

It uses quantum-inspired optimization algorithms and heuristics to underpin QUASAR's quantum optimization capabilities. QUASAR employs these techniques to optimize audit trail data's storage, retrieval, and processing, enabling efficient verification and analysis of large-scale audit trails. By leveraging quantum-inspired approaches, QUASAR can identify optimal data structures, indexing schemes, and verification strategies, reducing the computational overhead and storage requirements associated with audit trail management.

With its cryptographic analysis and integrity enforcement features, QUASAR monitors audit trails for any signs of tampering, inconsistencies, or anomalies. It employs advanced cryptographic techniques, such as zero-knowledge proofs and homomorphic encryption, to perform privacy-preserving analysis of the audit trails, detecting potential security breaches or integrity violations without revealing sensitive information.

QUASAR's architecture aligns with quantum-enhanced security, scalability, and efficiency principles. Its robust validation mechanisms, which include quantum-based penetration testing, cryptanalysis, and formal verification, ensure the resilience and effectiveness of the quantum-enhanced audit trail verification and protection mechanisms.

5 Quantum Computing implemented by QUASAR

QUASAR utilizes advanced quantum computing techniques, post-quantum cryptography, and quantum-inspired optimization to enhance the security, scalability, and efficiency of audit trail verification and protection.

10 Quantum Verification: QUASAR leverages quantum parallelism and quantum algorithms to perform efficient and scalable integrity and consistency verification of audit trails. The quantum verification process includes the following steps:

Audit Trail Encoding: QUASAR encodes relevant audit trail data, such as cryptographic signatures, data lineage, and temporal information, into quantum states using techniques like quantum amplitude encoding or quantum phase encoding.

15 Quantum Algorithm Design: The QuantumVerifier agent designs and implements quantum algorithms tailored specifically for audit trail verification, including quantum search algorithms, quantum walk algorithms, or quantum machine learning algorithms.

20 Quantum Circuit Compilation: QUASAR compiles the designed quantum algorithms into optimized quantum circuits, considering the specific quantum hardware architecture and limitations.

Quantum Execution: The quantum circuits are executed on a quantum computer or simulator, utilizing quantum parallelism to verify multiple aspects of the audit trails concurrently.

25 Result Interpretation: The quantum measurement results are interpreted and analyzed to determine the integrity and consistency of the audit trails, identifying any discrepancies or anomalies.

30 The time complexity of the quantum verification process may depend on the size of the audit trails, the complexity of the employed quantum algorithms, and the efficiency of the quantum hardware or simulator used. The space complexity may be determined by the number of qubits required to encode the audit trail data and the depth of the quantum circuits.

Post-Quantum Cryptography: QUASAR employs post-quantum cryptographic algorithms and digital signatures to safeguard the audit trails against potential threats posed by quantum computers. The post-quantum cryptography process includes the following steps:

Algorithm Selection: QUASAR selects suitable post-quantum cryptographic algorithms, such as lattice-based cryptography, multivariate cryptography, or hash-based signatures, based on their security properties and performance characteristics.

Key Generation: Using the selected algorithms, the PostQuantumSecurer agent generates the necessary post-quantum cryptographic keys, including public-private key pairs or signature keys.

Cryptographic Primitives: QUASAR implements the requisite post-quantum cryptographic primitives, such as encryption, decryption, signing, and verification, using the generated keys.

Audit Trail Protection: QUASAR utilizes post-quantum cryptographic primitives to safeguard audit trails, ensuring their long-term integrity and non-repudiation properties, even in the presence of quantum adversaries.

Security Analysis: QUASAR conducts ongoing security analysis and monitoring of post-quantum cryptographic mechanisms, keeping abreast of the latest advancements and potential vulnerabilities in the field.

The time complexity of the post-quantum cryptography process may be contingent upon the specific algorithms chosen, the sizes of critical elements utilized, and the efficiency of cryptographic primitives. The space complexity may be determined by the size of post-quantum cryptographic keys and any additional storage required for the protected audit trails.

Quantum-Inspired Optimization: QUASAR employs quantum-inspired optimization techniques to enhance the storage, retrieval, and processing of audit trail data, facilitating efficient verification and analysis of extensive audit trails. The process of quantum-inspired optimization entails the following steps:

Problem Formulation: QUASAR formulates the optimization problem for audit trail management as a mathematical optimization task, considering storage efficiency, retrieval latency, and verification performance.

Quantum-Inspired Algorithm Design: The QuantumOptimizer agent designs and tailors quantum-inspired optimization algorithms, such as quantum annealing, quantum-inspired

evolutionary algorithms, or quantum-inspired swarm intelligence, to address the formulated optimization problem.

Classical Execution: The quantum-inspired optimization algorithms are executed on classical computing hardware, leveraging the principles of quantum-inspired computing to explore vast solution spaces efficiently.

Solution Evaluation: The generated optimization solutions are assessed based on predefined performance metrics, such as storage overhead, retrieval time, or verification accuracy, to identify the most promising strategies.

Iterative Refinement: QUASAR iteratively improves the optimization solutions by incorporating feedback and domain-specific knowledge, continually enhancing the efficiency and effectiveness of audit trail management processes.

The time complexity of the quantum-inspired optimization process may depend on the scale and complexity of the audit trail data, the number of optimization variables and constraints, and the convergence properties of the employed quantum-inspired algorithms. The space complexity may be determined by the size of the optimization solution representations and any intermediate data structures required during the optimization process.

Cryptographic Analysis and Integrity Enforcement: QUASAR engages in ongoing cryptographic analysis and integrity enforcement to identify and mitigate tampering, inconsistencies, or anomalies in the audit trails. The process of cryptographic analysis and integrity enforcement encompasses the following steps:

Cryptographic Primitive Selection: QUASAR employs the selection of appropriate cryptographic primitives, including zero-knowledge proofs, homomorphic encryption, or secure multiparty computation, based on the security and privacy requirements of the audit trail analysis.

Audit Trail Analysis: The CryptographicAnalyzer agent employs the chosen cryptographic primitives to conduct a privacy-preserving analysis of the audit trails, examining the cryptographic signatures, data lineage, and temporal consistency while preserving the confidentiality of sensitive information.

Anomaly Detection: QUASAR utilizes machine learning algorithms and statistical techniques to identify anomalies, inconsistencies, or patterns that may indicate potential tampering or integrity violations within the audit trails.

Integrity Verification: The IntegrityEnforcer agent performs integrity verification of the audit trails by comparing the cryptographic analysis results with the expected values, ensuring that the audit trails remain unmodified and uncompromised.

Alert and Response: In the event of integrity violations or anomalies being detected,

- 5 QUASAR generates alerts and initiates appropriate response mechanisms, such as revoking compromised cryptographic keys, isolating affected segments of the audit trails, or initiating further investigations.

- 10 The time complexity of the cryptographic analysis and integrity enforcement process may vary depending on factors such as the volume of audit trail data, the complexity of the cryptographic primitives utilized, and the efficiency of the anomaly detection and integrity verification algorithms. The space complexity is determined by the size of the cryptographic analysis results, the anomaly detection models, and any temporary data structures employed during the analysis process.

QUASAR Design – Business Layer

- 15 Processes:

Quantum-Enhanced Audit Trail Verification: This process involves encoding audit trail data into quantum states, designing and executing quantum algorithms to efficiently verify integrity and consistency, and interpreting quantum measurement results to identify discrepancies or anomalies.

- 20 Post-Quantum Cryptographic Protection: This process focuses on selecting and implementing post-quantum cryptographic algorithms and digital signatures to safeguard audit trails against potential threats from quantum computers. It ensures the long-term security and non-repudiation properties of the audit trails.

- 25 Quantum-Inspired Optimization of Audit Trail Management: This process involves formulating audit trail optimization problems, designing and adapting quantum-inspired optimization algorithms, executing them on classical hardware, and iteratively refining optimization solutions to enhance the efficiency and effectiveness of audit trail storage, retrieval, and processing.

- 30 Continuous Cryptographic Analysis and Integrity Enforcement: This process focuses on selecting and applying appropriate cryptographic primitives to analyze audit trails in a privacy-preserving manner. It detects anomalies and inconsistencies indicative of tampering or integrity violations and triggers alerts and response mechanisms to ensure the ongoing integrity and security of the audit trails.

Functions:

Quantum-Accelerated Audit Trail Verification: QUASAR leverages quantum parallelism and quantum algorithms to perform efficient and scalable audit trail integrity and consistency verification. This significantly reduces the verification time compared to classical approaches.

Post-Quantum Cryptographic Resilience: QUASAR employs post-quantum cryptographic algorithms and digital signatures to protect the audit trails against potential threats from quantum computers. It ensures the long-term security and non-repudiation properties of the audit trails.

Quantum-Inspired Optimization of Audit Trail Efficiency: QUASAR utilizes quantum-inspired optimization techniques to optimize the storage, retrieval, and processing of audit trail data. This enables efficient verification and analysis of large-scale audit trails while reducing computational overhead and storage requirements.

Proactive Integrity Monitoring and Enforcement: QUASAR performs continuous cryptographic analysis and integrity enforcement. It detects tampering, inconsistencies, or anomalies in the audit trails and triggers alerts and response mechanisms to maintain the ongoing integrity and security of the audit trail data.

QUASAR Design – Application Layer

The QUASAR system comprises the following components:

QuantumVerifier: This component utilizes quantum parallelism and quantum algorithms to efficiently and scalably verify the integrity and consistency of audit trails. It encodes audit trail data into quantum states and executes customized quantum circuits.

PostQuantumSecurer: This component selects and implements post-quantum cryptographic algorithms and digital signatures to safeguard the audit trails against potential threats posed by quantum computers. It ensures long-term security and non-repudiation properties.

QuantumOptimizer: This component employs quantum-inspired optimization techniques to optimize the storage, retrieval, and processing of audit trail data. It formulates optimization problems and adapts quantum-inspired algorithms for efficient audit trail management.

CryptographicAnalyzer: This component continually analyzes the cryptographic aspects of the audit trails. It employs privacy-preserving cryptographic primitives to examine the integrity and consistency of the audit trail data without disclosing sensitive information.

IntegrityEnforcer: This component ensures the integrity of the audit trails by detecting anomalies, inconsistencies, or patterns that may indicate tampering or integrity violations. It triggers alerts and response mechanisms to maintain the ongoing security and reliability of the audit trails.

5 Services:

QuantumVerificationService: Provides methods for encoding audit trail data into quantum states, designing and executing quantum algorithms for efficient verification of integrity and consistency, and interpreting the results of quantum measurements.

10 PostQuantumSecurityService: Offers services for selecting and implementing post-quantum cryptographic algorithms and digital signatures to protect the audit trails against potential threats posed by quantum computers.

QuantumOptimizationService: This service enables the formulation of audit trail optimization problems, the design and adaptation of quantum-inspired optimization algorithms, and the execution of these algorithms on classical hardware for efficient audit trail management.

15 CryptographicAnalysisService: Facilitates the selection and application of cryptographic primitives for privacy-preserving analysis of audit trails. It detects anomalies and inconsistencies that may indicate tampering or integrity violations.

IntegrityEnforcementService: Provides services for enforcing the integrity of the audit trails, detecting anomalies and inconsistencies, and triggering alerts and response mechanisms to
20 maintain their ongoing security and reliability.

Interfaces:

QuantumVerifierInterface: Defines the methods and parameters for encoding audit trail data into quantum states, designing and executing quantum algorithms for verification, and interpreting the results of quantum measurements.

25 PostQuantumSecurerInterface: This interface specifies the methods and input/output formats for selecting and implementing post-quantum cryptographic algorithms and digital signatures to protect audit trails.

QuantumOptimizerInterface: This interface describes the methods and parameters for formulating audit trail optimization problems, designing and adapting quantum-inspired
30 optimization algorithms, and executing these algorithms on classical hardware.

CryptographicAnalyzerInterface: This interface defines the methods and input/output formats for selecting and applying cryptographic primitives to preserve privacy, analyze audit trails, and detect anomalies and inconsistencies.

IntegrityEnforcerInterface: Specifies the methods and parameters for enforcing the integrity of the audit trails, detecting anomalies and inconsistencies, and triggering alerts and response mechanisms.

QSEM and CATM - Quantum-Resilient Security and Auditable Data Provenance: QSEM's integration of quantum encryption algorithms and post-quantum cryptography can protect the confidentiality and integrity of financial crime data, ensuring long-term security against emerging threats from quantum computing. CATM's ability to generate immutable audit trails and cryptographically capture data provenance can provide a tamper-evident record of financial crime data, supporting robust investigations, regulatory audits, and forensic analysis.

By integrating these components, multi-agent neural architecture (such as GenFoundry) can create horizontal capabilities for financial crime that span data governance, narrative intelligence, access control, security, and auditing. This holistic approach can empower financial institutions to detect, investigate, and report financial crimes more effectively while maintaining robust data integrity, regulatory compliance, and operational resilience.

The quantum computing framework described herein is designed with technical advantages that extend beyond conventional natural language processing tasks. By employing a standardized and efficient method for quantum-optimized processing of language model outputs, the framework enables a range of applications, from advanced search and recommendation systems to automated content generation and comprehensive knowledge discovery. This facilitates enterprises and researchers to extract insights and generate knowledge from the extensive datasets of textual data.

As described herein, the framework's processing of human language leads to improvements in areas such as machine translation, sentiment analysis, and text summarization. Through the application of quantum computing, the framework identifies complex semantic relationships and captures contextual nuances, generating text that is both coherent and contextually relevant, and surpassing the output capabilities of traditional methods.

As described herein, the framework may be configured to be foundational to quantum-enhanced conversational AI systems, enabling human-machine interactions. By processing and understanding the nuances of human language more effectively, the framework supports the creation of intelligent virtual assistants, chatbots, and dialogue systems capable of interpreting context, resolving ambiguities, and delivering tailored responses.

The quantum computing framework described herein further advances the processing and enhancement of large language model outputs using ISO 11179 compliant JSON representations. By leveraging the capabilities of quantum computing and standardizing data handling techniques, the framework improves language understanding, generation, and analysis. It establishes a basis for intelligent language systems that efficiently process large amounts of unstructured textual data, extracting insights and producing coherent and contextually accurate text.

The quantum computing framework configured for processing and enhancing large language model outputs with a JSON format that adheres to a metadata registry standard may provide broad industrial applicability across diverse sectors. This framework significantly enhances the capacity to efficiently process, analyze, and derive insights from the extensive unstructured textual data produced by advanced language models, thereby serving critical roles in natural language processing, data analytics, decision support systems, and knowledge management.

Healthcare and Life Sciences: The framework may analyze electronic health records, medical literature, clinical trial data, and patient feedback, leveraging quantum-enhanced NLP techniques to extract insights crucial for medical decision-making. This includes identifying disease patterns, predicting patient outcomes, accelerating drug discovery, and refining treatment protocols.

Finance and Banking: Utilized by financial institutions to process extensive textual data such as news articles, market reports, and customer feedback, the framework may support the detection of market trends, assessment of investment risks, enhancement of fraud detection systems, and improvement of customer service through quantum-enhanced sentiment analysis and entity recognition.

E-commerce and Retail: The framework may be applied to customer reviews, product descriptions, and user-generated content to understand consumer preferences, optimize product recommendations, and personalize user experiences. Quantum-powered text classification, clustering, and summarization aid in categorizing products and identifying market trends.

Customer Service and Support: Integrated within customer service systems like chatbots and virtual assistants, the framework may enhance response accuracy and relevance, utilizing quantum-enhanced NLP to boost the efficiency of customer support operations, reduce response times, and increase customer satisfaction.

Cybersecurity and Threat Intelligence: The framework may process threat intelligence reports and security logs to identify potential threats and detect anomalies, supporting proactive cyber defense operations. Quantum-enhanced techniques aid in recognizing malicious actors and uncovering cyber-attack patterns.

- 5 Marketing and Social Media Analytics: Applied to the massive amounts of data generated by social media platforms, the framework may analyze user-generated content to identify trending topics, monitor brand sentiment, and refine marketing strategies. Quantum-powered sentiment analysis and opinion mining may provide real-time insights into user preferences.

- 10 Legal and Intellectual Property: The framework may be used to analyze legal documents, patents, and contracts, facilitating legal research and decision-making. Quantum-enhanced text analysis and knowledge graph techniques streamline the identification of key entities and relationships in legal texts.

- 15 Government and Public Sector: Employed by government agencies to process public records and policy documents, the framework may enhance public services and supports evidence-based policymaking. Quantum-enhanced text analysis helps categorize documents and identify key issues, optimizing resource allocation.

- 20 Education and Research: In academic settings, the framework may analyze scholarly articles and student feedback, assisting researchers in quickly identifying relevant information and uncovering new connections between concepts, facilitated by quantum-enhanced summarization and information retrieval.

Media and Content Analysis: Used by media organizations to analyze news articles and social media content, the framework may assist in understanding audience preferences and identifying trending topics through quantum-enhanced sentiment analysis and topic modeling.

- 25 The quantum computing framework may provide numerous extensions beyond its capabilities of processing and enhancing large language model outputs. These variations leverage quantum algorithms and circuits to expand the framework's functionality across various specialized tasks:

- 30 Quantum-Enhanced Text Generation: This extension may incorporate quantum algorithms designed for generating text, enhancing the diversity, coherence, and contextual relevance of outputs. Techniques such as quantum language modeling, quantum sampling, and quantum reinforcement learning are employed to elevate the creativity and expressiveness of language models.

Quantum-Optimized Data Compression: The framework may integrate quantum-based data compression techniques to efficiently store and transmit ISO 11179 compliant JSON representations. Techniques like quantum source coding, quantum data deduplication, and quantum entropy coding significantly reduce storage and bandwidth requirements, optimizing the management of large-scale textual data.

Quantum-Enhanced Multilingual Processing: Adaptations of the framework may support multilingual output from language models by utilizing quantum algorithms designed for cross-lingual NLP tasks. Quantum machine translation, quantum cross-lingual information retrieval, and quantum language transfer learning enhance the accuracy and efficiency of processing multilingual textual data.

Quantum-Assisted Knowledge Graph Construction: The framework can be adapted to include quantum algorithms for building and querying knowledge graphs from language model outputs. Quantum graph embedding, quantum graph traversal, and quantum knowledge representation learning enable efficient and precise extraction of entities, relationships, and semantic information.

Quantum-Enhanced Sentiment Analysis: The framework can be adapted to include quantum circuits for sentiment analysis, improving the accuracy and depth of emotional content analysis in text. Quantum sentiment classification, quantum emotion detection, and quantum affective computing deepen the insights into the affective aspects of textual data.

Quantum-Optimized Named Entity Recognition: Incorporating quantum algorithms optimized for named entity recognition enhances the identification and classification of entities such as persons, organizations, and locations, improving precision and recall in entity extraction.

Quantum-Enhanced Text Summarization: Quantum circuits can be designed for text summarization tasks enable the generation of concise and informative summaries. Quantum sentence compression, quantum topic modeling, and quantum abstractive summarization provide more coherent and relevant textual summaries.

Quantum-Assisted Question Answering: The framework can be enhanced with quantum algorithms for question answering, improving the accuracy and efficiency of information retrieval from text. Techniques like quantum information retrieval, quantum semantic search, and quantum reading comprehension facilitate faster and more precise answers.

Integration with Quantum NLP Frameworks: The framework can be integrated with existing quantum NLP frameworks such as QNLP and QNLU, leveraging their quantum algorithms

and libraries to enhance capabilities and foster advanced quantum-enhanced NLP applications.

Quantum-Enhanced Interpretability and Explainability: By incorporating quantum algorithms, the framework may improve the interpretability and explainability of language model outputs, providing clearer insights into the decision-making processes behind AI-generated text, thus enhancing trust and accountability.

Quantum-Optimized Language Model Fine-tuning: Extensions of the framework may include quantum algorithms for fine-tuning language models to specific domains or tasks.

Techniques like quantum transfer learning, quantum meta-learning, and quantum few-shot learning offer more efficient and effective model adaptation.

Quantum-Accelerated Language Model Training: Incorporating quantum algorithms to accelerate the training phase of language models reduces computational complexity and resource requirements, allowing for the development of more advanced models.

Quantum-Enhanced Multimodal Processing: The framework can handle multimodal inputs by integrating quantum circuits for processing text, images, and audio. Quantum multimodal attention, cross-modal retrieval, and representation learning improve the accuracy and robustness of multimedia data processing.

Quantum-Optimized Language Model Evaluation: Quantum techniques may be employed to evaluate and benchmark the performance of language models, providing scalable and accurate methods for assessing model quality and supporting deployment decisions.

Quantum-Accelerated Language Model Inference: The framework may leverage quantum algorithms to enhance the speed and efficiency of language model inference, facilitating more responsive and real-time NLP applications.

In example embodiments, a Quantum Natural Language Processing (QNLP) system, or CODA natural language data is first processed by the Quantum Data Representation Module, where it is transformed into a format amenable to quantum computations. This includes the generation of quantum embeddings via amplitude encoding techniques. Subsequently, the Quantum Processing Module employs specialized quantum circuits to perform advanced NLP tasks such as semantic analysis and text summarization, leveraging the enhanced processing capabilities of quantum mechanics. The integration of these processes is managed by the Quantum-Classical Integration Module, which ensures seamless data flow between classical and quantum computational resources, facilitating a hybrid computing approach that enhances both efficiency and output accuracy. This

approach represents a significant advancement in the field of natural language processing by incorporating quantum computing technologies.

5 The QNLP system may be configured to enhance the processing of natural language data through the integration of quantum computing techniques. It operates on data that includes domain labels and descriptions, utilizing quantum algorithms to perform various language processing tasks. The system may be structured to not only handle basic language processing but also to generate complex outputs that include metrics and assessments such as semantic similarity scores, concept confidence scores, and drift metrics among others.

Data Handling and Processing:

10 The QNLP system begins by processing structured text data, preparing it for further analysis. It categorizes this data by labels and numerically indexes it, setting the stage for more detailed processing.

Quantum-Enhanced Computation:

15 Utilizing quantum algorithms, the QNLP system computes embeddings for concepts within the text. These embeddings represent textual data in a quantum-friendly format, enabling the application of quantum computational advantages such as faster processing and handling of large datasets.

Advanced Data Analysis:

20 The QNLP system calculates how closely related various concepts within the data are, using quantum algorithms to evaluate semantic similarities more efficiently than classical methods might allow. Additionally, it employs reasoning and inference mechanisms to derive new knowledge from the processed data, further enriching the understanding and utility of the information.

Output Quality and Consistency Checks:

25 QNLP system assesses the consistency and reliability of outputs by comparing expected outputs with those generated under modified conditions (hallucinated outputs). This helps in identifying and measuring any deviations, which are critical for maintaining the accuracy of the system.

Decision Support:

30 By simulating different scenarios and calculating confidence scores for various concepts based on the processed data, the QNLP system aids in decision-making processes, highlighting the most relevant and reliable information.

Continuous Improvement:

The QNLP system updates its knowledge base with new information and user feedback, ensuring that the models it uses are continually refined and improved based on the latest data.

5 Operational Integration and Reporting:

After processing, the QNLP system is capable of saving detailed reports and results, facilitating easy integration into existing workflows and systems for further action or analysis.

Quantum Data Representation Module:

10 A Quantum Data Representation Module enables the conversion of language model outputs into a structured, quantum-ready format. In embodiments, the resulting converted dataset may adhere to an ISO 11179-compliant JSON representation. The Quantum Data Representation Module employs parsing, preprocessing, and encoding techniques to generate well-structured textual data that includes necessary quantum metadata. This quantum-enabled data representation allows the quantum computing framework to utilize
15 quantum algorithms and quantum circuits for efficient processing and extensive analysis of large-scale language data.

The Quantum Data Representation Module may be configured to prepare, encode, and evaluate language model outputs to facilitate efficient processing using quantum computing techniques. FIG. 4 is a flowchart 400 illustrating an example process implemented by the
20 Quantum Data Representation Module for preparing language model outputs for quantum computing, according to an embodiment herein.

In step 402, the output data from one or more language models is analysed to extract a dataset. This initial step involves the parsing of raw textual outputs to extract relevant information, such as sentences, paragraphs, named entities, and semantic relationships. For
25 example, the outputs may be parsed to extract plain text, embeddings, or sequences generated by the language model, which are useful for subsequent quantum processing.

In step 404, the extracted dataset is prepared for compatibility with a standardized data format and quantum computing. In this step, the extracted data may be prepared by employing techniques such as tokenization, normalization, and embedding generation to
30 ensure the data meets standards required for both quantum computing and structured data formats. This pre-processing step may include normalizing the data to align it with JSON formatting requirements and quantum computing specifications.

In step 406, the prepared dataset is encoded into a structured dataset, wherein the structured dataset includes metadata for quantum computing operations. In this step, the prepared data may be then encoded into an ISO 11179 compliant JSON format. This JSON structure may be specifically crafted to encompass intricacies and interdependencies inherent in natural language, while also integrating quantum-specific metadata fields. For example, the pre-processed and normalized data from step 404 may be encoded into JSON structures that comply with the ISO 11179 standard, incorporating essential metadata elements such as qubit count and gate operations, for enabling quantum computing operations by the quantum processing module.

In step 408, the structured dataset is evaluated for compatibility with a metadata standard. After the encoding of step 406, the dataset may be evaluated to ensure it meets specific metadata standards. This evaluation verifies that the dataset's structure and metadata are correctly implemented, supporting both the integrity and functionality of subsequent quantum processing. For example, the generated JSON representations may be validated against the ISO 11179 standard to verify data quality, consistency, and allow for interoperability across different systems.

Parsing and Extraction

The Quantum Data Representation Module may employ natural language processing (NLP) libraries to parse and extract relevant information from the raw text data. For example, the NLP libraries may include Natural language Toolkit (NLTK) or spaCy. The NLP pipeline for parsing and exaction may include a series of steps that start with tokenization, where the text may be split into individual elements such as words and punctuation. This can be achieved using rule-based or statistical tokenizers. Following tokenization, each token may be subjected to Part-of-Speech (POS) tagging, where tokens may be labeled with tags indicating their syntactic roles, such as nouns, verbs, or adjectives. This process may employ pre-trained POS taggers that utilize either machine learning models or rule-based approaches. As the analysis progresses, Named Entity Recognition (NER) may be applied to identify and classify named entities like people, organizations, and locations using models often based on conditional random fields or advanced neural networks like BERT.

Dependency Parsing may further dissect the text to outline its syntactic structure, identifying relationships among words such as subject, object, and modifier relationships. This step may employ dependency parsers grounded in machine learning or graph-based algorithms. Following Dependency Parsing, Semantic Role Labeling (SRL) may detect the roles words or phrases in a sentence, delineating the predicate-argument structure of sentences using techniques like support vector machines or deep neural networks. Coreference Resolution

may be employed to establish links between mentions of the same entity across the text, thus resolving ambiguities and establishing clear coreference chains. This can be achieved through the application of rule-based systems, machine learning models, or their combination. The extracted information includes sentences, paragraphs, named entities, semantic relationships, dependencies, and other relevant linguistic structures.

Pre-processing

Once the relevant information has been extracted, the Quantum Data Representation Module may employ a series of preprocessing techniques to normalize and prepare the data for subsequent processing. Initially, text may be converted to lowercase to ensure consistency across the dataset. In embodiments, punctuation marks can be omitted, as they may be determined to be not relevant to certain analytical tasks and may disrupt the formatting process of process 300. Additionally, stopwords—common words such as "the," "is," and "at" that typically possess little semantic value—may be eliminated using pre-defined stopwords lists to streamline the text. Following these initial steps, lemmatization may be applied, transforming words into their base or dictionary forms (lemmas). Lemmatization facilitates in reducing the number of inflectional forms in the text, thereby improving the generalization capabilities of the subsequent analyses. In order to capture deeper semantic and contextual meanings, word embeddings may be generated using techniques such as Word2Vec or GloVe. These techniques represent words as dense vectors within a high-dimensional space. If the word embeddings are pre-trained, they may be fine-tuned on domain-specific language model outputs to enhance their ability to capture relevant semantics and context, thereby further customizing the process to meet the specific requirements of the given task.

Encoding

Once the data has been pre-processed, it may be encoded into a JSON format that adheres to the ISO 11179 metadata registry, specifically designed to encapsulate the complex linguistic structures and relationships inherent in the text. The JSON structure utilized may be a nested JSON object, systematically organizing fields to capture the raw text, sentences, paragraphs, and other extracted information. Named entities such as people, organizations, and locations may be encoded as key-value pairs, where the key indicates the entity type, and the value provides the entity's text, position in the text, and other pertinent metadata. Additionally, semantic relationships, dependencies, and other extracted linguistic structures may be represented as nested JSON objects or arrays, effectively capturing the essential relationships between entities or linguistic components. Furthermore, word embeddings or other contextual embeddings derived from models like BERT may be encoded as arrays or

tensors within the JSON structure. This process of embedding and encoding preserves comprehensive semantic and contextual information of the words, ensuring that the encoded data retains its linguistic intricacy and usefulness for further computational tasks.

Qubit Connectivity Optimization:

- 5 The Quantum Data Representation Module may analyse the tensor representations of language data for superconducting quantum processors with limited qubit connectivity, such as nearest-neighbor architectures. The Quantum Data Representation Module may perform tensor reshaping, compression, and reorganization of the qubit mapping to minimize the number of SWAP gates required when executing on the hardware. For all-to-all connected
10 architectures, the Quantum Data Representation Module can exploit denser, higher-dimensional tensor representations.

Quantum Error Models:

- The Quantum Data Representation Module may be designed to manage quantum error models effectively. The Quantum Data Representation Module may characterize the noise
15 processes, coherence times, and gate fidelities of the target quantum processor, tailored to the specific hardware type—such as superconducting, ion-trap, or photonic. Based on this characterization, the Module may integrate suitable quantum error correction codes, dynamical decoupling sequences, and noise mitigation strategies into the JSON encoding metadata. For topological quantum processors, which may have inherent fault tolerance, the
20 encoding can be specifically tailored to exploit these properties.

Discrete vs Analog Quantum Models:

- The Quantum Data Representation Module may differentiate between discrete and analogue quantum models to further optimize data representation. For gate-model quantum processors, the Quantum Data Representation Module may optimize tensor factorizations
25 and encodings to efficiently reuse qubit registers and reduce swap overhead when mapping to quantum circuits. In contrast, for analogue quantum simulators and quantum annealers, the Quantum Data Representation Module may adapt the embeddings to map linguistic constraints onto the natively supported Hamiltonian forms, ensuring compatibility and maximizing performance.

30 Quantum Processor Calibrations:

The Quantum Data Representation Module may constantly monitor and update quantum processor calibration data, such as qubit coherence times, gate fidelities, optimal operation frequencies, etc. Moreover, the Quantum Data Representation Module may fine-tune the

parsing/embedding pipelines and incorporate these hardware-specific calibrations into the JSON metadata for optimal performance.

Quantum Control Optimizations:

Based on available quantum control resources, the Quantum Data Representation Module may tailor parsing or embedding strategies for processors with programmable quantum control planes. For example, ion-trap quantum computers can provide exceptional control over motional degrees of freedom, which can be leveraged via tensor reformulations.

In implementations, the Quantum Data Representation Module may employ automated mechanisms and algorithms to determine optimal values for quantum-specific metadata fields like qubitCount, gateSet, and connectivityGraph based on the attributes of the input language data.

Qubit Count Determination:

The Quantum Data Representation Module may begin the optimization process by analysing the tensor representations and dimensions of the encoded language data. The Quantum Data Representation Module may utilize techniques such as tensor rank analysis and matrix bipartitioning to estimate the minimum number of qubits necessary for the storage and processing of this data on quantum processors. For example, if the language embedding forms a high-dimensional but sparse tensor, methods such as Tensor Train decompositions may be employed to ascertain the required qubit footprint. The qubit count may be subsequently established as this minimum number, with allowances for auxiliary qubits for purposes such as error correction and temporary storage.

Gate Set Selection:

The gate selection algorithm may begin by examining the structure, sparsity patterns, and multilinear ranks of the language tensors. This analysis aids in identifying which gates from the universal gate set (such as H, X, Y, Z, CNOT) are used for executing core tensor operations. The algorithm may prioritize low-rank factorizations for sparse language tensors, which require only a subset of the complete gate set. Additionally, gate selection may be tailored to accommodate the connectivity constraints and topological properties of the target quantum processor, ensuring optimal operation.

Connectivity Graph Generation:

The optimization process encompasses determining an optimal qubit layout and mapping of language tensors to enhance a reuse of qubit registers and minimize the requirement for SWAPs. Subsequently, the module generates connectivity graph metadata, which outlines how qubits encoding related linguistic features should be interconnected. Techniques such

as tensor canonical and train decompositions may be employed to identify subsets of qubits that necessitate specific connectivity, such as CNOT or SWAP links. The tensor layouts may be subsequently adjusted to align with the processor's connectivity limitations, providing a balance between qubit reuse and SWAP overheads.

- 5 In implementations, the Quantum Data Representation Module may integrate an automated framework and algorithms for the purpose of dynamically modifying and revising the metadata fields that are specific to quantum computing technology and hardware requirements. This automated framework may utilize several techniques:

Quantum Hardware Abstraction Layer:

- 10 The Quantum Hardware Abstraction Layer may facilitate in concealing intricate low-level hardware details of various quantum processor architectures behind a uniform interface. This layer provides a standardized set of quantum computational primitives or operations, which the pipeline's parsing, encoding, and metadata generation processes can utilize. As advancements in quantum hardware occur, the capabilities of new quantum processors may
15 be integrated into this abstraction. This integration allows for the continuous adaptation and refinement of algorithms to align with the evolving capabilities of quantum computing technologies.

Quantum Processor Benchmarking:

- The Quantum Data Representation Module may include a benchmarking engine designed to
20 continuously evaluate the performance of the latest quantum processors using representative language processing tasks and datasets. This benchmarking helps identify potential quantum advantages and hardware-specific issues such as bottlenecks, cooling constraints, and topological restrictions. The insights derived from these benchmarks guide necessary adjustments in data mapping, qubit allocation, connectivity graphs, and gate
25 selections to enhance performance and efficiency on new quantum hardware platforms.

Multi-Objective Optimization Algorithms:

- Integrated in the Quantum Data Representation Module may be multi-objective optimization algorithms that govern core parsers, encoders, and tensor mapping processes. These algorithms are designed using formulations that adapt to changes in objective functions or
30 constraints, employing advanced optimization techniques such as quantum approximate optimization, tensor network optimization, and symbolic manipulation as meta-optimizers. When new hardware primitives become available, these optimization objectives can be updated to re-optimize tensor encodings and refine metadata fields.

Online Learning and Feedback Loop:

An online learning engine may be integrated within the automated framework to monitor and analyse the actual performance of language processing tasks on newly deployed quantum hardware. In embodiments, the engine may compare real-time operational data against theoretical predictions derived from earlier benchmarking to identify any performance discrepancies. Subsequently, a feedback loop may be activated, which updates the parsers, encoders, and embedding strategies based on the gathered performance data, allowing the automated framework to dynamically adjust to the evolving capabilities of emerging quantum processors.

Quantum Control Plane Integration:

The automated framework may interface with the calibration and configuration systems of quantum processors that feature programmable control planes. This integration allows the metadata generation algorithms to automatically adapt to changes in control parameters, such as qubit operating points and gate fidelity optimums. These adaptations may be driven by updates and advancements in hardware technologies, ensuring that the module's operations are optimized for the latest generation of quantum processors.

In addition to complying with the ISO 11179 standard for metadata representation, the design of the Quantum Data Representation Module may also consider other relevant international standards and regulations to ensure compliance and interoperability. One such standard is ISO/IEC 5116:

ISO/IEC 5116: This international standard, titled "Information technology—Programming languages, their environments and system software interfaces—Data interchange representations for data sharing," specifies data interchange representations for sharing data among programming languages, environments, and platforms. The ISO/IEC 5116 standard defines syntax, semantics, and encoding specifications for data representations that can be used for data interchange.

The design of the module may incorporate aspects of ISO/IEC 5116 in the following manner:

Data Interchange Formats: The JSON-based encoding used to represent the language data may adhere to the syntax and encoding rules specified in ISO/IEC 5116. This ensures that quantum-optimized data representations can be seamlessly exchanged and interpreted across different programming languages, platforms, and quantum computing environments.

Semantic Interoperability: ISO/IEC 5116 establishes mechanisms to ensure the preservation of data semantics during interchange. The module's encoding strategies may effectively

capture and retain the extensive semantic information present in the language data, thereby enabling precise interpretation and processing of quantum systems.

Extensibility: The ISO/IEC 5116 standard provides guidelines for extending data interchange representations to accommodate domain-specific requirements. The Quantum Data Representation Module's JSON encoding may follow these guidelines, allowing the incorporation of quantum-specific metadata fields while maintaining compatibility with existing data interchange tools and frameworks.

Platform Independence: By adhering to ISO/IEC 5116, the data representations of the module remain platform-independent, enabling their usage across diverse classical and quantum computing environments, programming languages, and operating systems.

Validation and Conformance Testing: The ISO/IEC 5116 standard provides methods for validating data interchange representations and verifying their conformance to the specified rules. The Quantum Data Representation Module may incorporate validation mechanisms to ensure that the generated JSON encodings comply with both ISO 11179 and ISO/IEC 5116 standards.

ISO 20022 is an internationally recognized standard for electronic data interchange among financial institutions. The ISO 20022 standard provides a universal language and approach for describing financial business processes, data, and messaging. The Quantum Data Representation Module can incorporate ISO 20022 in the following ways:

Financial Data Representation: When working with financial data, the Quantum Data Representation module can leverage ISO 20022 message formats and data structures to represent financial transactions, accounts, and other relevant information. This provides interoperability with existing financial systems and compliance with industry standards.

Semantic Mapping: The Quantum Data Representation module can establish a mapping between the semantic concepts in the financial domain (as defined by ISO 20022) and the quantum-optimized data representations. This mapping enables smooth integration and interpretation of financial data within the quantum computing framework.

Metadata Alignment: The metadata fields of the Quantum Data Representation module can be aligned with the relevant metadata elements specified in ISO 20022. This includes transaction timestamps, financial institution identifiers, and message types, ensuring consistency and compatibility with established financial industry practices.

There are several ISO standards specifically addressing quantum computing technologies and their applications. Some pertinent standards are as follows:

ISO/IEC 80000-13:2008: This standard, entitled "Quantities and units — Part 13: Information science and technology," establishes quantities and units relevant to information science and technology, including those pertaining to quantum computing. The Quantum Data Representation module may utilize the terminology and unit conventions provided by this standard when describing quantum-specific metadata fields.

ISO/IEC TR 23690:2020: This technical report, entitled "Information technology — Quantum computing — Terminology and vocabulary," presents a comprehensive collection of terms and definitions related to quantum computing. The Quantum Data Representation module may adopt the terminology and vocabulary established in this standard to ensure clarity and consistency in communication and documentation.

ISO/IEC 23837:2021: This standard, entitled "Information technology — Security techniques — Security requirements, test and evaluation methods for quantum key distribution," focuses on the security aspects of quantum key distribution (QKD) systems. Although not directly relevant to data representation, the Quantum Data Representation module may consider the security guidelines and evaluation methods outlined in this standard when processing sensitive financial data and ensuring the overall security of the quantum computing environment.

ISO/IEC 15909 is an international standard that establishes a general-purpose graphical notation for Petri nets. Petri nets are a mathematical modelling language used to describe and analyse concurrent, distributed, and parallel systems. The standard is comprised of three parts:

ISO/IEC 15909-1: Systems and software engineering — High-level Petri nets — Part 1: Concepts, definitions, and graphical notation.

ISO/IEC 15909-2: Systems and software engineering — High-level Petri nets — Part 2: Transfer format.

ISO/IEC 15909-3: Systems and software engineering — High-level Petri nets — Part 3: Extensions and structuring mechanisms.

The Quantum Data Representation module can utilize ISO/IEC 15909 in the following ways:

Modelling Quantum Processes: Petri nets can be used to model and analyse the behaviour of quantum algorithms and quantum circuits. The Quantum Data Representation module can employ the graphical notation and concepts defined in ISO/IEC 15909-1 to visually represent and document the quantum processes involved in data representation and processing.

Petri Net Transfer Format: ISO/IEC 15909-2 specifies a transfer format for exchanging Petri net models between different tools and environments. The Quantum Data Representation module can employ this transfer format to import or export Petri net models of quantum processes, enabling interoperability with other Petri net-based tools and frameworks.

- 5 Extensions and Structuring: ISO/IEC 15909-3 defines extensions and structuring mechanisms for Petri nets, such as hierarchical structuring, time concepts, and stochastic extensions. The Quantum Data Representation module can leverage these extensions to model and analyse more complex quantum processes, incorporating timing constraints, probabilistic behaviour, and hierarchical structures.

- 10 Through the incorporation of ISO/IEC 15909 into the Quantum Data Representation Module, several advantages can be achieved:

Visual Modelling: Petri nets offer an intuitive, standardized graphical notation for modelling and communicating quantum processes. This visual representation enhances comprehension and collaboration among stakeholders involved in the development and
15 analysis of quantum systems.

- Formal Analysis: Petri nets have a robust mathematical foundation, enabling the use of formal analysis techniques such as reachability analysis, deadlock detection, and performance evaluation. By utilizing Petri nets to model quantum processes, the module can utilize these analysis techniques to verify and validate the accuracy and efficiency of
20 quantum data representation and processing.

Interoperability: Adhering to the ISO/IEC 15909 standard ensures that the Petri net models used in the Quantum Data Representation Module are compatible with other Petri net-based tools and frameworks. This promotes interoperability and seamless integration with existing modelling and analysis environments.

- 25 Extensions and Flexibility: The extensions and structuring mechanisms provided by ISO/IEC 15909-3 enable the Quantum Data Representation Module to model and analyse more complex quantum processes, considering timing constraints, probabilistic behaviour, and hierarchical structures. This flexibility enhances the module's capability to accurately capture and represent real-world quantum systems.

- 30 By incorporating these additional ISO standards, the Quantum Data Representation Module provides several advantages:

Financial Industry Compliance: Adhering to ISO 20022 enables the Quantum Data Representation Module to integrate with existing financial systems, processes, and messaging protocols, thereby facilitating its adoption and usage in the financial industry.

Quantum Computing Alignment: By adhering to quantum computing ISO standards, the Quantum Data Representation Module may align with industry best practices, terminology, and conventions, thereby enhancing its credibility and interoperability within the quantum computing community.

Security Considerations: Incorporating security standards like ISO/IEC 23837 helps address security challenges associated with quantum computing, particularly when handling sensitive financial data.

Interoperability and Consistency: Adhering to these ISO standards promotes interoperability, data consistency, and semantic clarity across different quantum computing platforms, programming languages, and financial systems.

By designing the Quantum Data Representation module in compliance with ISO 11179, ISO/IEC 5116, ISO 20022, and relevant quantum computing ISO standards, the module provides a standardized and industry-aligned approach to representing and processing financial data in quantum computing environments.

The Quantum Data Representation Module enhances the scalability and efficiency of the framework by effectively managing the exponential growth of language model outputs. The compact and structured JSON format enables efficient storage, retrieval, and transmission of large amounts of textual data, thereby addressing storage and bandwidth challenges associated with unstructured text. Additionally, the JSON representation allows for rapid parsing and processing by quantum algorithms, facilitating scalability of computational operations as the volume of language data increases.

Compliance with the ISO 11179 standard brings benefits in terms of data governance, interoperability, and reusability. The standardized metadata framework promotes data consistency, quality, and semantic interoperability across quantum computing platforms and applications. It establishes a common vocabulary and structure for describing and exchanging language data, fostering collaboration and knowledge sharing within the quantum computing community. ISO 11179 compliance also enables smooth integration of the framework with existing data management systems and tools, facilitating the incorporation of quantum-optimized language processing capabilities.

To accommodate the unique requirements of different quantum computing architectures and algorithms, the Quantum-Optimized Data Representation module offers a flexible and

extensible design. The JSON structure can easily incorporate additional metadata fields, allowing for the inclusion of algorithm-specific parameters, error correction schemes, and hardware-specific optimizations. This adaptability ensures that the framework can evolve and leverage advancements in quantum computing technologies and techniques.

- 5 The efficacy and efficiency of the Quantum-Optimized Data Representation module can be validated through rigorous performance benchmarking and comparative analysis. Metrics such as data compression ratios, encoding/decoding speeds, and memory utilization can be assessed to demonstrate the superiority of the JSON-based representation over traditional unstructured text formats. The modularity of the framework enables integration with classical data storage and retrieval systems, utilizing industry-standard APIs and protocols to facilitate an exchange of language data between classical and quantum computing environments.

The design choices and implementation of the Quantum Data Representation Module may be based on extensive research and best practices from the fields of quantum computing, metadata management, and natural language processing. The use of JSON as a lightweight and versatile data interchange format has been widely adopted in various domains, showcasing its effectiveness in handling complex data structures and promoting interoperability.

By leveraging the advantages of the Quantum Data Representation Module, the quantum computing framework establishes a basis for efficient and scalable processing of language model outputs using quantum computing techniques. The Quantum Data Representation Module enables integration with quantum algorithms and circuits, facilitates data governance and reusability, and establishes groundwork for the development of advanced quantum-enhanced natural language processing applications.

In embodiments, the Quantum Data Representation Module may be configured to integrate with existing classical computing frameworks through a sophisticated quantum-classical integration layer. The quantum-classical integration layer facilitates efficient interoperability between quantum algorithms that operate on encoded language data and classical data processing pipelines. It supports transition and functionality across both computing paradigms.

In embodiments, the Quantum Data Representation Module may be configured to process the transfer of large volumes of language data between classical storage solutions like data lakes and quantum processors, providing efficient data ingress and egress. This configuration facilitates seamless data flow necessary for integrating classical and quantum computing environments. The Quantum Data Representation Module may use high-bandwidth, low-latency data transfer protocols such as NVLink, GPUDirect, and InfiniBand.

Additionally, the Quantum Data Representation Module may be capable of streaming data ingest directly from classical language model APIs and services, ensuring a continuous flow of data into the quantum system.

Quantum-Specific Metadata Fields

- 5 The JSON representation may contain quantum-specific metadata fields that enable efficient integration with quantum computing systems and algorithms. These quantum-specific metadata fields may include:

qubitCount: The number of qubits required for the computation.

gateSet: The set of quantum gates available for the computation.

- 10 gateOperations: A list of quantum gate operations to be applied.

connectivityGraph: Information about the connectivity between qubits on a QPU device.

rabiFrequencyRange, detuningRange, phaseRange: Ranges for setting the Rabi frequency, detuning, and phase parameters when programming an analog Hamiltonian simulation.

deviceCost: The pricing information for using a particular quantum device.

- 15 drivingFields: For defining the time-dependent driving fields like amplitude, phase, detuning for an analog Hamiltonian simulation program.

shots: The number of times to repeat the circuit execution for sampling.

outputLocation: The storage endpoint and prefix for storing the results.

deviceParameters: Device-specific parameters like qubit layout, noise models etc. action:

- 20 The type of quantum computation task (e.g. circuit model, analog simulation etc.)

paradigm: The quantum computing paradigm like gate-based, analog Hamiltonian etc.

The quantum-specific metadata fields enable quantum algorithms and circuits to efficiently process and analyze the language model outputs. For example, the qubitCount field specifies the number of qubits required for the computation, allowing the quantum system to
25 allocate the necessary resources. This may be determined by analysing the dimensionality and complexity of the encoded data using techniques like tensor rank analysis or matrix bi-partitioning algorithms. The qubit count may be selected to ensure compact representation while minimizing the required number of qubits.

The gateSet field specifies the set of quantum gates needed to manipulate and process the encoded data on a quantum computer. The choice of gate set may be determined by analyzing the tensor representations of the encoded data, their sparsity patterns, and the types of operations required for natural language processing tasks. Common gate sets include Pauli rotations (X, Y, Z), Hadamard gates (H), and controlled gates (CNOT, CCNOT). More complex gate sets may be chosen based on the complexity of the language model outputs and the specific quantum algorithms used.

The gateOperations field contains a list of quantum gate operations that can be applied to the encoded data on a quantum computer. The specific gate operations may be determined by analysing the tensor representations of the encoded data and the quantum algorithms designed for natural language processing tasks. The gate operations may be optimized to minimize circuit depth, reduce the number of SWAP operations required (based on the target quantum hardware's qubit connectivity), and ensure efficient execution on the target quantum processor.

The connectivityGraph field represents the connectivity constraints of the target quantum hardware, specifying how qubits are connected and can interact with each other. The connectivity graph may be determined based on the hardware specifications of the target quantum processor (e.g., superconducting qubits with nearest-neighbour connectivity, ion traps with all-to-all connectivity). The encoded data and gate operations may be optimized to minimize the overhead of SWAP operations required to satisfy the connectivity constraints.

The errorCorrectionCode field specifies the quantum error-correcting code (QECC) to be used to protect the encoded data from decoherence and errors during quantum computations. The selection of the QECC may depend on factors such as the noise characteristics of the target quantum hardware, the complexity of the encoded data, and the level of fault tolerance required. Common QECCs include the Steane, Shor, surface, and topological codes.

The deviceParameters field encompasses various device-specific parameters that may be considered for efficient execution on the target quantum hardware. These parameters may include qubit layout, noise models, calibration data, gate fidelities, coherence times, and other hardware-specific constraints or characteristics. These parameters optimize the execution of quantum circuits on the target device and mitigate the effects of hardware imperfections. The rabiFrequencyRange, detuningRange, and phaseRange fields specify the ranges for setting the Rabi frequency, detuning, and phase parameters, respectively, when programming an analog Hamiltonian simulation, allowing for fine-tuned control over the quantum system. The deviceCost field provides pricing information for using a particular

quantum device, enabling cost-effective resource allocation and budgeting for quantum computations. The drivingFields field defines the time-dependent driving fields, such as amplitude, phase, and detuning, for an analog Hamiltonian simulation program, allowing for precise control over the quantum system's evolution. The shots field specifies the number of times to repeat the circuit execution for sampling, enabling the collection of statistical data and improving the accuracy of quantum computations. The outputLocation field designates the storage endpoint and prefix for storing the results, facilitating efficient data retrieval and analysis. The action and paradigm fields specify the type of quantum computation task and the quantum computing paradigm, respectively, allowing for the selection of appropriate quantum algorithms and circuits based on the specific requirements of the language processing task.

The algorithms utilized for parsing and extraction, preliminary processing, encoding, and quantum-specific metadata computations may be formulated to enhance the representation of language data for optimal processing on quantum computers. These procedures may use NLP techniques, tensor analysis methodologies, and principles of quantum computing to ensure that the encoded data can be efficiently manipulated and processed using quantum algorithms and circuits specifically designed for natural language processing tasks.

The development of data and reporting systems for financial regulations can benefit from using the methodologies in the quantum computing framework. These methodologies enable the parsing, extraction, pre-processing, and encoding of financial data into a format that may be tailored for quantum computing. Quantum-specific metadata calculations enhance the efficiency with which financial data may be represented on quantum computers. This increased efficiency enables faster and more complex financial simulations, risk analyses, and regulatory compliance assessments. Additionally, the processed financial data can be consolidated in a centralized repository, facilitating easy retrieval, analysis, and reporting. The organized format and integrated metadata may further streamline querying, aggregation, and visualization of financial data, thereby facilitating regulatory reporting and compliance oversight. A detailed explanation of the specific algorithms used is provided below:

During the Parsing and Extraction phase, Natural Language Processing (NLP) techniques may be utilized to extract pertinent information from financial documents such as regulatory filings, prospectuses, and financial reports. Named entity recognition may be employed to identify key entities, such as company names, financial metrics, dates, and monetary amounts. Additionally, dependency parsing and relation extraction techniques may be

applied to determine semantic relationships between these entities, such as the associations between companies and their financial metrics.

In the Pre-processing stage, text normalization techniques may be applied to standardize the extracted financial information. This may include converting numbers to a consistent format, normalizing date formats, and handling abbreviations and acronyms specific to the financial domain. Moreover, embeddings for financial terms and entities may be generated using techniques such as Word2Vec or GloVe, which are trained on a large corpus of financial documents. These embeddings help capture the semantic similarities and relationships among financial concepts.

- 10 The Encoding process may include converting the extracted and pre-processed financial information into a structured format that adheres to financial reporting standards such as JSON or XML, aligning with XBRL or ISO 20022. A hierarchical structure may be utilized to represent the relationships between financial entities such as companies, financial statements, and individual data points. Additional metadata, including data lineage, data quality scores, and data provenance, may be encoded to ensure the traceability and reliability of the financial data.

- 20 In the Quantum-Specific Metadata Calculations, a qubit count may be determined based on the complexity and volume of the financial data. This determination may include analysing the dimensionality of the financial data points and estimating the number of qubits required to efficiently represent them. A specific gate set may be defined based on the financial calculations and transformations required, which may include gates for arithmetic operations, financial functions like present value and amortization, and statistical computations. The gate set and connectivity constraints may be subsequently optimized to align with the available quantum hardware and the specific financial algorithms to be implemented.

- 25 The parsing, pre-processing, and encoding algorithms implemented in the quantum computing framework may be tailored to accommodate distinctive requirements and limitations of quantum computing systems. The following are some factors that may be considered:

- 30 Tensor Representations: The process of encoding language data into tensor representations suitable for quantum computing can be achieved through the following steps: Firstly, a parsing process may extract semantic relationships, entities, and other elements from the language data and organizes them into structured tensors, rather than just simple word sequences. Subsequently, a pre-processing phase may generate high-

dimensional word/sentence embeddings. These embeddings can then be efficiently converted into quantum state vectors by employing techniques like amplitude encoding.

Quantum Parallelism: Traditional algorithms in the field of Natural Language Processing (NLP) process language in a sequential manner. However, by employing parsing and

- 5 encoding techniques that can extract semantic representations, quantum parallelism can be employed. In example embodiments, sentences or paragraphs may be represented as tensor networks, which enables the utilization of quantum tensor operations, such as contractions in superposition.

Qubit Efficiency: A number of qubits can be a significant limitation in NISQ devices. The

- 10 encoding process used by the data encoding module 104 may map the language data onto the fewest possible qubits by storing semantic features that are relevant to downstream quantum NLP tasks. This avoids storing the entire raw text, which would be inefficient in terms of qubit usage. Tensor decomposition techniques may assist in compressing the qubit usage.

15 Connectivity Constraints: When conducting JSON encoding, connectivity constraints imposed by quantum processors may be considered. This can be accomplished by appropriately reshaping tensors and utilizing methodologies such as SWAP insertion during circuit synthesis. By doing so, it is possible to circumvent the costly overhead associated with SWAP operations.

20 Quantum Error Handling: Decoherence effects may intensify with depth of circuits and the number of qubits used in quantum computations. In embodiments, the pre-processing phase may reduce circuit depth by embedding language into high-dimensional but sparse quantum states. Quantum error correction and fault-tolerant techniques can be incorporated into the JSON encoding metadata.

25 Quantum Advantage Analysis: A quantum advantage analysis module examines the complexity and structure of language data to determine whether mapping the language data to quantum states offers advantages over classical methods. This analysis helps to determine an optimal partitioning of workloads between classical and quantum approaches.

The above pre-processing, parsing, and encoding algorithms reconfigure traditional

- 30 language representations into optimized versions for quantum computing primitives such as superposition, entanglement, quantum parallelism, and complexity theory, while adhering to quantum constraints. This facilitates the utilization of quantum advantages for language processing.

FIG. 5 is a block diagram illustrating an example quantum computing framework 500 for processing unstructured input data from language models through a series of quantum and classical modules, according to an embodiment herein.

Figure 5 illustrates an integrated workflow for processing unstructured input data within

5 quantum computing framework 500, emphasizing the dynamic interaction between quantum and classical processing modules. Unstructured input data from large language models 502 is initially pre-processed by the Quantum Data Representation Module 504. The Quantum Data Representation Module 504 prepares the data in a format that is suitable for quantum processing. Subsequently, the formatted data is input into the Quantum Processing Module
10 506 where the formatted data is subjected to quantum computations designed to leverage properties of quantum mechanics for enhanced processing efficiency. Following quantum computing, the data outputted from the Quantum Processing Module 506 may be sent to the Quantum-Classical Integration Module 508, which may determine an appropriate processing pathway for the data outputted. In embodiments, this processing pathway may be
15 determined via a task scheduler incorporated in the Quantum-Classical Integration Module 508 such as a Quantum Advantage Determination Engine (QUADE). QUADE may dynamically evaluate whether subsequent processing corresponding to the processed output data from the Quantum Processing Module 506 should continue within the quantum domain or sent to a Classical Processing Module 510 for classical processing. This evaluation by the
20 Quantum-Classical Integration Module 508 may be based on complexity analysis, quantum resource estimation, and quantum noise simulation, ensuring that each task is handled efficiently. This functionality by the Quantum-Classical Integration Module 508 is depicted by the bi-directional arrows, indicating that data output from the Quantum Processing Module 506 can be forwarded to the Classical Processing Module 510 for classical computing or
25 sent back to the Quantum Processing Module 506 for additional quantum processing-based assessments by the task scheduler of the Quantum-Classical Integration Module 508. Results from quantum computations may be integrated into classical systems with error mitigation and fault tolerance strategies applied to ensure data integrity.

The Classical Processing Module 510 may further refine data output from the Quantum-

30 Classical Integration Module 508 for practical applications within classical computing environments. Notably, feedback from the classical processing module 510 to the Quantum-Classical Integration Module 508 may enable further refinements, leveraging real-time feedback to adjust processing strategies or re-engage quantum computations as needed.

This iterative feedback loop is crucial for continuously improving the accuracy and efficiency
35 of the data output. This workflow 500 effectively utilizes the strengths of both quantum and classical computing paradigms, facilitated by the intelligent routing capabilities of the task

scheduler, ensuring optimal computational efficiency and high accuracy in the processed outputs, while maintaining the flexibility to adapt to changing data conditions and processing requirements.

The quantum computing framework described herein may be configured to be hardware-agnostic and adaptable to different types of quantum processors and their unique characteristics. The framework may incorporate several techniques to optimize the parsing, encoding, and metadata generation processes based on particulars of the quantum hardware platform.

The quantum computing framework and other techniques described herein may be designed with technical advantages, including scalability, adaptability, and interoperability. The modular structure of such framework allows for integration of emerging quantum algorithms and circuits, keeping the framework aligned with advancements in quantum natural language processing. In examples, utilization of an ISO 11179 compliant JSON format standardizes data representation and supports the exchange of language data across various quantum computing platforms and programming environments, enhances the framework's applicability in diverse technological contexts.

FIG. 6 depicts a schematic representation of a computing device 600, designed to operate with both classical and quantum computing processes of the quantum computing framework 500, in accordance with the embodiments of the systems and methods disclosed herein. The device comprises a central processing unit (CPU) 622, which interfaces with various data storage and memory components: secondary storage 624, read-only memory (ROM) 626, and random-access memory (RAM) 628.

Within this configuration, a Quantum Computing Unit (QCU) 636 is integrated. The QCU 636 utilizes quantum mechanical phenomena to enhance computation. It is engineered to execute algorithms that are particularly suitable for quantum computation, such as those involving large-scale number factorization, quantum simulations, and specific optimization problems, enabling computational performance that exceeds the capabilities of a standalone CPU.

The secondary storage 624 may include a dedicated sector 624a containing instructions executable by both the CPU 622 and the QCU 636. These instructions enable the device 1600 to perform operations that may be optimized through quantum computing.

The ROM 626 stores immutable code essential for an initial booting process and routine operations of the computing device 600. The RAM 628 provides volatile memory for immediate access to data by the CPU 622 and the QCU 636 during active tasks.

Peripheral devices are managed through input/output (I/O) interfaces 630, while network connectivity is enabled via a network interface 632. A graphics processing unit (GPU) 634 is present to handle parallel processing tasks, which may be separate from or integrated with quantum computing processes.

- 5 The CPU 622, as the primary processor for general computing tasks, is responsible for executing sequential operations and handling a variety of computational processes in the classical domain. It manages routine tasks with high efficiency and interfaces with the system's memory, including RAM 628 and ROM 626, for data storage and retrieval.

10 Within this dual-capability system, the CPU 622 often acts as a coordinator, determining when to engage the QCU 636 based on the computational requirements. For quantum-suitable tasks, the CPU 622 prepares and relays data to the QCU 636, which then processes this information utilizing its quantum computing power. After the QCU 636 completes the quantum processing, it can transmit the results back to the CPU 622, which may perform additional classical processing or output the results.

- 15 The collaboration between the CPU 622 and the QCU 636 effectively expands the device's 600 computational range, enabling it to switch between classical and quantum operations. This ensures that the device 600 utilizes the most efficient processing method available, whether that be the classical computation provided by the CPU 622 or the quantum processing offered by the QCU 636.

20 Although the computing device 600 is described with reference to a single computer, it should be appreciated that the computing device may be formed by two or more computers in communication with each other that collaborate to perform a task. For example, but not by way of limitation, an application may be partitioned in such a way as to permit concurrent and/or parallel processing of the instructions of the application. Alternatively, the data
25 processed by the application may be partitioned in such a way as to permit concurrent and/or parallel processing of different portions of a data set by the two or more computers. In an embodiment, virtualization software may be employed by the computing device 600 to provide the functionality of a number of servers that is not directly bound to the number of computers in the computing device 600. In an embodiment, the functionality disclosed above may be
30 provided by executing the application and/or applications in a cloud computing environment. Cloud computing may comprise providing computing services via a network connection using dynamically scalable computing resources. A cloud computing environment may be established by an enterprise and/or may be hired on an as-needed basis from a third-party provider.

Additional components, such as one or more application specific integrated circuits, neuromorphic computing units, field programmable gate arrays, or other electronic or photonic processing components can also be included and used in conjunction with or in place of the processor 622 to perform processing operations. The processing operations can include machine learning operations, other operations supporting the machine learning operations, or a combination thereof.

The technical solution detailed in present disclosure may be embodied in the form of a computer program product. The computer program product may be stored in a non-volatile or non-transitory storage medium, which can be a compact disk read-only memory, USB flash disk, or a removable hard disk. The computer program product includes a number of instructions that enable a computer device (personal computer, server, or network device) to execute the methods provided in the embodiments described herein. For example, such an execution may correspond to a simulation of the logical operations, including the training and aggregation of model updates in the federated learning process, as described herein. The software product may additionally or alternatively include number of instructions that enable the computing device 600 to execute operations for configuring or programming a digital logic apparatus in accordance with embodiments of the present invention.

By programming and/or loading executable instructions onto the computing device, at least one of the CPU 622, the RAM 628, and the ROM 626 are changed, transforming the computing device in part into a specific purpose machine or apparatus having the novel functionality taught by the present disclosure. It is fundamental to the electrical engineering and software engineering arts that functionality that can be implemented by loading executable software into a computer can be converted to a hardware implementation by well-known design rules.

It will be appreciated to those skilled in the art that the preceding examples and embodiments are exemplary and not limiting to the scope of the present disclosure. Whilst the foregoing description has described exemplary embodiments, it will be understood by those skilled in the art that many variations of the embodiment can be made within the scope and spirit of the present invention. It shall be noted that elements of any claims may be arranged differently including have multiple dependencies, configurations and combinations.

Claims:

1. A system for data processing, comprising:

a classical data interface configured to receive data from a classical computing environment;

5 a data encoding module configured to encode the received data into a format compatible with a quantum computing environment;

a quantum processing module within the quantum computing environment configured to process the encoded data;

10 a communication interface configured to transfer the encoded data to the quantum computing environment and retrieve the processed data from the quantum computing environment; and

a data decoding module configured to decode the processed data into a format compatible with the classical computing environment.

15 2. The system of claim 1, wherein the data encoding module is further configured to transform the received data into a structured dataset that is compatible with a metadata standard.

3. The system of claim 2, wherein the structured dataset that is compatible with a metadata standard corresponds to an ISO 11179 compliant JSON representation.

20 4. The system of any one of the preceding claims, wherein the quantum processing module is configured to use at least one quantum algorithm selected from the group comprising of quantum support vector machines (QSVMs) and quantum neural networks (QNNs).

25 5. The system of any one of the preceding claims, further comprising a task scheduler for allocating tasks between the classical computing environment and quantum computing environment based on at least one of the following criteria: complexity of the data, suitability of processing requirements for quantum or classical computation, and available computational capacity in both computing environments.

6. A method for data processing, comprising:

receiving data from a classical computing environment;

30 encoding the received data into a format compatible with a quantum computing environment using a data encoding module;

transferring the encoded data to the quantum computing environment;

processing the encoded data in the quantum computing environment using a quantum processor;

retrieving the processed data from the quantum computing environment; and

5 decoding the processed data into a format compatible with the classical computing environment using a data decoding module.

7. The method of claim 6, wherein the decoded processed data is further integrated into downstream applications in the classical computing environment.

8. The method of claim 6 or 7, further comprising applying a plurality of task scheduling
10 algorithms within the quantum computing environment based on characteristics of quantum processors.

9. The method of any one of claims 6-8, wherein the encoded data transferred to the quantum computing environment and the processed data decoded for the classical computing environment are both in a structured data format that is compatible with a
15 metadata standard.

10. The method of any one of claims 6-9, wherein the method is implemented as part of a hybrid quantum-classical computing system that supports a hardware-agnostic abstraction layer which allows the computing system to interact with a plurality of quantum processors.

11. A method for transforming language model outputs, comprising:

20 parsing an output dataset corresponding to one or more language models to extract a dataset;

processing the extracted dataset to generate a refined dataset;

encoding the refined dataset into a structured dataset including metadata parameters, wherein the structured dataset is configured for quantum processing; and

25 evaluating the structured dataset for compatibility with a plurality of quantum computing systems.

12. The method of claim 11, wherein the metadata parameters comprise at least one of qubit requirements, quantum gate operations, and connectivity between qubits and a quantum processing unit.

13. The method of claim 11 or 12, wherein the structured dataset configured for quantum processing corresponds to an international protocol specific to quantum computing, high-level petri nets, or information technology.

14. The method of any one of the preceding claims, wherein the structured dataset is ISO 11179 compliant for providing compatibility with a plurality of quantum computing systems.

15. A method for processing language model outputs, comprising:

analysing output data from one or more language models to extract a dataset;

preparing the extracted dataset for compatibility with a standardized data format and quantum computing;

10 encoding the prepared dataset into a structured dataset, wherein the structured dataset includes metadata for quantum computing operations; and

evaluating the structured dataset for compatibility with a metadata standard.

16. A system for quantum natural language processing, comprising:

15 a quantum computing device configured to execute quantum circuits for processing quantum states;

a memory storing instructions executable by a processor to:

encode input data into the quantum states using quantum circuits executed by the quantum computing device;

20 apply one or more quantum operations via the quantum circuits to the encoded quantum states for processing at least one natural language task selected from the group comprising of semantic analysis, sentiment classification, named entity recognition, and text summarization;

25 measure the encoded quantum states after the application of the one or more quantum operations to determine a result of the processing of the at least one natural language task; and

output the determined result in a structured dataset format suitable for classical data processing.

17. The system of claim 16, wherein the one or more quantum operations include a Quantum Latent Semantic Analysis (QLSA) to identify semantic relationships within the text data.

5 18. The system of claim 16 or 17, wherein the quantum operations include at least one of a Quantum Support Vector Machine (QSVM), a Quantum Generative Adversarial Network (QGAN), or a Quantum Variational Autoencoder (QVAE).

19. The system of any one of claims 16-18, wherein the quantum operations include at least one of quantum entanglement and superposition principles to enhance processing speed for the at least one natural language task.

10 20. The system of any one of claims 16-19, wherein the structured dataset format is ISO 11179 compliant for providing compatibility with a plurality of quantum computing systems.

21. The system of any one of claims 16-20, wherein the input data is provided as a structured dataset that is compatible with a metadata standard prior to encoding of the input data.

15 22. The system of any one of claims 16-21, wherein the quantum operations include a quantum text summarization algorithm configured to generate summaries of text by using at least one of quantum phase estimation and quantum amplitude amplification.

20 23. The system of any one of claims 16-22, wherein the structured dataset format includes metadata corresponding to at least one of a natural language task, a quantum operation applied, a device parameter, and a number of quantum measurements.

24. The system any one of claims 16-23, wherein the quantum operations include the use of Hadamard and controlled gates within the quantum circuits to encode a plurality of sentiment features into the quantum states.

25

30