



Quantum machine learning for natural language processing application

Shyambabu Pandey, Nihar Jyoti Basisth, Tushar Sachan, Neha Kumari, Partha Pakray*

Computer Science & Engineering, NIT Silchar, India

ARTICLE INFO

Article history:

Received 29 April 2023

Received in revised form 9 July 2023

Available online 6 August 2023

Keywords:

Quantum computing

Quantum machine learning

Natural language processing

POS tagging

ABSTRACT

Quantum computing is a speedily emerging area that applies quantum mechanics properties to solve complex problems that are difficult for classical computing. Machine learning is a sub-field of artificial intelligence which makes computers learn patterns from experiences. Due to the exponential growth of data, machine learning algorithms may be insufficient for big data, whereas on other side quantum computing can do fast computing. A combination of quantum computing and machine learning gave rise to a new field known as quantum machine learning. Quantum machine learning algorithms take advantage of the fast processing of quantum computing and show speedup compared to their classical counterpart. Natural language processing is another area of artificial intelligence that enables the computer to understand human languages. Now, researchers are trying to take advantage of quantum machine learning speedup in natural language processing applications. In this paper, first, we discuss the path from quantum computing to quantum machine learning. Then we review the state of the art of quantum machine learning for natural language processing applications. We also provide classical and quantum-based long short-term memory for parts of speech tagging on social media code mixed language. Our experiment shows that quantum-based long short-term memory performance is better than classical long short-term memory for parts of speech tagging of code-mixed datasets.

© 2023 Elsevier B.V. All rights reserved.

1. Introduction

Quantum computing [1] is a branch of computer science that applies a different computing paradigm based on the principle of quantum mechanics [2]. Unlike classical computing, which uses binary representation for information, quantum computing uses quantum bit (Qubit) that can exist in multiple states simultaneously. Quantum algorithms like Shor's prime factorization algorithms [3] and Grover's search algorithm [4] show the power of quantum computing. The prime factorization problem requires billions of years by classical systems, while Shor's algorithms solve it within a few hours. Grover's search algorithm searches an element in an unsorted database, which shows quadratic speedup compared to its classical counterpart.

Quantum computing is capable of solving problems that are difficult for classical computing. It follows the principles of quantum theory like superposition, entanglement [5], and interference [6] that allow computation more efficiently than classical computation. Superposition enables a qubit to store all the possible values of the quantum state at a time. This

* Corresponding author.

E-mail address: partha@cse.nits.ac.in (P. Pakray).

property of superposition creates exponentially sizable computational space. Entanglement is another property that tries to connect the multiple quantum states and grow the computation speed exponentially. Another property is interference, which plays a vital role in quantum-based algorithms. It increases the probability of the desired outcome and reduces the likelihood of the non-desired outcome of algorithms. These properties make quantum computing a different paradigm of computing that can solve complex problems more efficiently. Quantum algorithms perform better than their classical counterpart using these properties. This opens up new possibilities for solving complex problems in fields such as machine learning [7].

Machine learning (ML) is a branch of artificial intelligence [8] that allow computers to learn from experiences without being explicitly programmed. ML algorithms aim to identify patterns from data and make decisions according to those patterns. ML is widely applied in engineering and science fields like natural language processing [9], image processing [10], and computer vision [11]. Other ML applications are spam email filtering, classification of text data, and finding the customer's behavior in the financial sector. In short, we can use ML algorithms where the computer can interpret the data based on an experience.

In the last 30 years, due to data availability and computational resources, ML algorithms achieved impressive success in various fields. It has become a very popular part of computer science to apply in multiple domains to solve problems. However after a couple of years, the increment of data is much greater than classical computer's performance growth. The exponentially growing data limit the computational power of classical ML. On the other hand, quantum computing can perform high-speed computing. So, researchers want to combine quantum computing with ML algorithms to take leverage of quantum computing speedup.

A joining of quantum computing and machine learning algorithms is known as Quantum machine learning (QML) [12]. QML is a field where the power of quantum computing is used to generate quantum versions of ML algorithms. The idea behind QML is to use the unique properties of quantum computing to develop new algorithms and models for ML that are more efficient and powerful than classical ML algorithms.

Quantum computers can perform faster computations and learn complex patterns from fewer data. QML algorithms can provide various advantages by using quantum computing's principle. So, QML comes up with benefits like improving the run time (giving quick results), advanced learning efficiency (by using fewer amount of training data to learn complex patterns), and advanced learning ability (increase in the capacity of content-addressable memory) [13]. Considering the advantages of QML algorithms, we can apply them in natural language processing.

Natural language processing (NLP) is another branch of artificial intelligence that enables the computer to understand and generate human languages. ML algorithms and computational linguistics are used to analyze, model, and develop natural languages. NLP has a wide range of applications like machine translation [14], text-summarization [15], information retrieval [16], and question answering [17]. NLP has mainly three fundamental processing methods: lexical, syntactic, and semantic analysis. Lexical analysis is used to divide the entire chunk of text into smaller units for further processing. Next, syntactic analysis is used to identify the structure of sentences (order of words and relationships between words in a sentence). The semantic analysis aims to determine the meaning of the text and the importance of each word in the given context of sentences.

Parts of speech (POS) [18] tagging is a fundamental task for several NLP applications. It automatically assigns a parts-of-speech tag to every word in a text. POS tagging involves using computational algorithms to analyze the context and meaning of each word in a text to determine its part-of-speech label. There are several different methods and algorithms used for POS tagging, including rule-based methods [19], probabilistic methods [20], and neural network-based methods [21].

Today the era of social media is used for sharing information, communication, marketing and advertising, and education and learning. Most of the time, social media text uses code-mixed languages, which use multiple languages. Processing social media code-mixed language is essential for several reasons: understanding user behavior, improving multilingual communication, and improving machine translation.

This paper discusses the background of quantum computing and quantum machine learning. We have discussed the state-of-the-art work of QML for NLP applications. We perform POS tagging on the code-mixed dataset using quantum-based long short-term memory (QLSTM) and classical long short-term memory (LSTM). We have preprocessed the dataset and split it into a fixed size of different batches for the experiments. Our experiments show that the result of QLSTM is better than classical LSTM for Facebook (Hindi-English, Telugu-English), WhatsApp (Telugu-English), and Twitter (Hindi-English, Bengali-English).

The paper is structured as Section 2 discusses the fundamentals of quantum computing, and Section 3 gives the background knowledge of QML. In Section 4, we provide the literature review related to QML, quantum neural networks, and the use of QML for NLP applications. Section 5 explains the different tools and platforms for QML. Section 6 gives the database description, Section 7 describes the setup of the experiment, and Section 8 shows the result of the experiment. Section 9 explains the error analysis, and Section 10 concludes the paper with a conclusion.

2. Quantum computing

Any computing model performs three tasks map inputs into outputs, store the information, manipulate the data by operators, and extract the result. Quantum computing is a computing model, like other computing models, based on the principles of quantum physics.

2.1. Quantum bit (qubit)

Qubit is a fundamental unit for carrying information in quantum computing. It is the quantum analog of classical bits, a basic unit of information in classical computing. The classical bit has a state of either 0 or 1 just like that qubit has two basis states $|0\rangle$ and $|1\rangle$, which is represented by Dirac notation ($|\rangle$). Unlike classical bits, qubits can form a state from the linear combination of basic states called superposition states.

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (1)$$

or

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (2)$$

or

$$|\psi\rangle = a|0\rangle + b|1\rangle = \begin{bmatrix} a \\ b \end{bmatrix} \quad (3)$$

where a and b are complex number, and $|a|^2 + |b|^2 = 1$. After measuring $|\psi\rangle$, it will collapse to basis state $|0\rangle$ with probability $|a|^2$ and $|1\rangle$ with $|b|^2$ probability. A quantum system may contain many qubits, so combining individual single qubit states is required to construct a compound state. The tensor product (\otimes) is used to construct the combined state of given individual single qubit states. For example, two quantum states $|\psi\rangle = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}$ and $|\psi'\rangle = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$, the tensor product of quantum states can describe their collective states.

$$|\psi\psi'\rangle = |\psi\rangle \otimes |\psi'\rangle = \begin{bmatrix} a_0 \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \\ a_1 \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} a_0b_0 \\ a_0b_1 \\ a_1b_0 \\ a_1b_1 \end{bmatrix} = a_0b_0|00\rangle + a_0b_1|01\rangle + a_1b_0|10\rangle + a_1b_1|11\rangle \quad (4)$$

where $a_0b_0, a_0b_1, a_1b_0, a_1b_1$ are complex numbers, and they must be normalized $|a_0b_0|^2 + |a_0b_1|^2 + |a_1b_0|^2 + |a_1b_1|^2 = 1$ because the outcome of measurement value must be equal to 1. Like two quantum states, multiple individual quantum states can be combined to build a compound quantum state.

2.2. Quantum operator

Quantum gates [22] are fundamental quantum operators to manipulate qubits. A quantum gate is a mathematical operation that acts on a qubit or a collection of qubits to perform a specific task. It is analogous to a classical logic gate in classical computing, which serves a boolean operation on classical bits while it performs a linear function on qubits. Quantum gates are reversible, so the number of inputs and output will be the same in any quantum circuit. Unitary matrices are used to represent the quantum gates that transfer one quantum state to another quantum state.

$$U|\psi\rangle = \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} = |\phi\rangle \quad (5)$$

Single qubit gates are known as fundamental gates of quantum computers, which are used to change a single qubit state to another quantum state. Table 1 shows some examples of single-qubit gates. Generally, multiple qubit gates are used in quantum algorithms to take quantum computing advantage, like entanglement, which connects various quantum states. These gates are capable of manipulating multiple qubits quantum systems. Table 2 shows examples of various qubits gates.

Parameterized quantum circuit (PQC) [23] is a fundamental building block for QML, quantum neural networks, and quantum optimization algorithms. These gates are parameterized with some parameter values, which are used to encode the classical data into a quantum state and train the quantum models. Table 3 shows examples of PQC gates.

2.3. Measurement

Measurement is a vital concept of quantum computing for extracting the classical information of quantum systems. When a quantum state is measured, it collapses into one possible outcome based on a certain probability. Quantum measurement operators measure quantum states. Suppose a quantum state $|\phi\rangle$ is measured in outcome m ; the following Eq. (6) determines the measurement probability.

$$P(m) = \langle \phi | M_m | \phi \rangle \quad (6)$$

Table 1

Examples of single qubit gate.





Gate name	Diagram	Matrix representation	Description
Pauli-X		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	It change $ 0\rangle$ to $ 1\rangle$ and $ 1\rangle$ to $ 0\rangle$, behave like classical Not gate.
Pauli-Y		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$	It rotate a quantum states to y-axis.
Pauli-Z		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	It changes the phase of a quantum state.
Hadamard		$\begin{bmatrix} \frac{1}{\sqrt{2}} & 0 \\ 0 & -\frac{1}{\sqrt{2}} \end{bmatrix}$	It convert a quantum states into superposition state.

Table 2

Examples of multiple qubit gate.




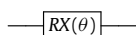
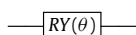
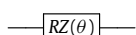
Gate name	Diagram	Matrix representation	Description
CNOT		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	If control qubit value is 1 then, it changes target value $ 0\rangle$ to $ 1\rangle$
Controlled-Z		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$	If control qubit value is 1 then, it changes phases of target value
Toffoli		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$	If first and second control bit is 1 then target value will be change.

Table 3

Examples of parameterized quantum gate.

Gate name	Diagram	Matrix representation	Description
RX		$\begin{bmatrix} \cos \theta/2 & -i \sin \theta/2 \\ -i \sin \theta/2 & \cos \theta/2 \end{bmatrix}$	It rotates the quantum states to x-axis by angle θ .
RY		$\begin{bmatrix} \cos \theta/2 & -\sin \theta/2 \\ \sin \theta/2 & \cos \theta/2 \end{bmatrix}$	It rotates the quantum states to y-axis by angle θ .
RZ		$\begin{bmatrix} \exp -i\theta/2 & 0 \\ 0 & \exp i\theta/2 \end{bmatrix}$	It rotates the quantum states to z-axis by angle θ .

3. Quantum machine learning

QML uses parameterized circuits to encode the classical data and train machine learning models. The first step in QML algorithms is encoding classical data into the quantum state because quantum computing cannot understand classical data. After encoding the classical data, QML models are implemented by a quantum circuit known as a variational quantum circuit. Fig. 1 shows the overview of VQC, where the first part $U(X)$ represents the encoding block and $U(\theta)$ represents ML models.

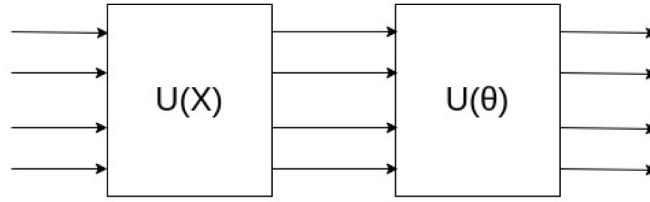


Fig. 1. An overview of VQC.

3.1. Encoding methods

Encoding methods are a crucial part of QML; they encode classical data x_1, x_2, \dots, x_n into quantum states, which can then be used in quantum algorithms. There are various encoding methods. Following are some popular encoding techniques used in QML algorithms.

Basis encoding [24] is a technique that encodes classical data into the basis states of a quantum system. In basis encoding, each classical data point is encoded into a set of qubits, and the basis states correspond to the possible values of the classical data point. This technique is often used for QML tasks that involve discrete data. But, this technique requires M qubits for each data M bit in the dataset. So, it does not apply to large and complex datasets.

$$|X\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^M |x^m\rangle \quad (7)$$

Amplitude encoding [25] is a simple encoding technique that encodes classical data into the amplitudes of quantum states. In amplitude encoding, each classical data point is encoded into a single qubit, and the amplitude of the qubit is set to correspond to the value of the classical data point. Where $N = 2^n$ is the element of and is the computational basis state. It represents a normalized classical N -dimensional data point, x , as the amplitudes of an n -qubit quantum state.

$$|\psi_x\rangle = \sum_{i=1}^N x_i |i\rangle \quad (8)$$

The advantage of amplitude encoding is it requires fewer qubits to encode. For example, suppose a dataset X contains m sample with each n -feature. To encode dataset X , we need one amplitude vector whose length is nm . A quantum system of N -qubit generates 2^n amplitude. So, It requires $N \geq \log nm$ qubits to encode n -dimensional data with m samples. The disadvantage is that the circuit for amplitude encoding is very deep because amplitude encoding encodes the entire dataset at a time, increasing the circuit's depth.

Angle encoding [26] is a more complex encoding technique that uses the angle to encode classical data into quantum states. It differs from the previous two encoding methods, as it only encodes one data point at a time rather than a whole dataset. For example, the data point can be encoded by the following equation, where N is the feature of the dataset.

$$|x\rangle = \bigotimes_{i=1}^N \cos x_i |0\rangle + \sin x_i |1\rangle \quad (9)$$

3.2. Variation quantum circuit

A variational quantum circuit [27] (VQC) is a type of quantum circuit that can be used to solve optimization problems and train QML models. VQC is a combination of quantum gates parameterized by a set of variables. The main goal of using VQC is used to find the optimal values of variables that minimize the cost function. The cost function can represent a wide range of optimization problems, such as finding the lowest energy state of a molecular system or optimizing the parameters of a machine learning model.

VQC can be trained by using a technique called quantum gradient descent, which is similar to classical gradient descent but uses quantum circuits to compute the gradients of the cost function for the variables [28]. The gradients are then used to update the values of the variables in each iteration of the optimization process. One of the advantages of VQC is its ability to use quantum computing to solve optimization problems intractable for classical computers.

There are also several challenges to developing practical QML algorithms. One of the main challenges is the high error rate of quantum hardware, which can lead to inaccurate results. There is ongoing research in developing error-correcting codes for quantum computers and hybrid classical-quantum machine learning algorithms that combine the strengths of classical and quantum computing. Overall, it is an exciting field with the potential to transform many areas of machine learning and artificial intelligence. However, much work still needs to be done to develop practical and scalable QML algorithms.

3.3. Quantum neural network

Another area of interest in QML is quantum neural networks [29], which use quantum circuits to perform operations on qubits in a way that mimics the behavior of classical neural networks. This approach can provide faster training and better performance for specific machine-learning tasks.

A quantum neural network (QNN) is a machine learning model that uses quantum computing principles to perform computations. The basic structure of a QNN is similar to that of a classical neural network, where layers of nodes or neurons are used to process input data and produce output. The neurons' weights in a QNN are implemented by parameterized quantum gates, which can perform complex transformations on the qubits.

QNN is a quantum analogous to classical neural networks. Classical neural networks are irreversible so that data can be represented as $x_1, x_2, \dots, x_n \rightarrow y$. But quantum mechanics is reversible, so the number of input and output are the same. Data can be represented $x_1, x_2, \dots, x_n, 0 \rightarrow x_1, x_2, \dots, x_n, y$ in QNN, where 0 is ancilla qubit. Learnable VQC can create the processing unit of QNN.

One of the challenges in developing QNN is the need for error correction, as noise in quantum hardware and decoherence. Researchers are exploring various techniques for error correction and developing algorithms that can effectively use the limited resources of quantum hardware. Overall, QNN represents an exciting area of research in quantum computing and machine learning, with the potential to revolutionize a wide range of applications in science and industry.

4. Literature review

Quantum computing and machine learning are exciting fields of this century. Machine learning algorithms make computers learn from experience or data. At the same time, quantum computing can solve complex problems efficiently. We can benefit from both fields using QML, combining quantum computing and machine learning. Here, we present previous work on QML, QNN, and QML algorithms for NLP applications.

HHL algorithm [30] is a quantum algorithm designed for solving linear equations that arise in most machine learning algorithms. HHL algorithm shows exponential speedup compared to its classical version under some assumptions. This algorithm is considered as starting point of QML. It can be used in data processing, machine learning, and numerical calculation.

The first phase of QML depends on linear algebra computation in high-dimensional spaces. QML algorithms like support vector machine, principle component analysis, and recommendation system [31] claimed exponential speedup compared to their classical counterpart. These algorithms require quantum random access memory (QRAM), while present quantum computers have no QRAM.

The next phase of QML algorithms wants to take advantage of current non-fault tolerance devices. It begins with hybrid quantum-classical algorithms [32], which combines classical and quantum algorithms. These algorithms are implemented by a family of VQC that are parameterized by some tunable parameters. Most of the present QML algorithms belong to this phase which can apply to the simulator and real quantum devices.

Researchers have designed many QML algorithms. They have compared the performance of these QML algorithms with their classical counterparts. The comparison of some QML algorithms has given in Table 4, which shows that few QML algorithms perform quadratic speedup, and few perform exponential speedup.

Hybrid classical-quantum algorithms develop a new area, QNN, the quantum version of the classical neuron network. These algorithms are best suitable for near-term quantum devices and try to implement these algorithms in various fields.

A quantum recurrent neural network (QRNN) [33] is a type of QNN model that is based on recurrent neural networks (RNNs) and uses quantum circuits to perform computations. Like classical RNN, QRNNs are designed to process sequential data and have feedback connections that allow them to retain information from previous time steps. However, instead of using classical computations to process the data, QRNN uses quantum circuits to perform computations on the data. In a QRNN, the input data is first encoded into quantum states and then fed into a quantum circuit, which computes the data. The output of the circuit is then measured and used as the input for the next time step. This process is repeated for each time step, allowing the QRNN to process sequential data.

A quantum convolutional neural network (QCNN) [34] is a type of QNN model that is based on the principles of classical convolutional neural networks (CNN). Like classical CNN, QCNN is designed to process data with a grid-like structure, such as images or spectrograms. However, QCNN uses quantum circuits to perform operations such as convolution, pooling, and activation, unlike classical CNN.

Quantum long short-term memory (QLSTM) [35] is a type of recurrent neural network (RNN) that uses quantum computing technology to perform computations. QLSTM is a quantum version of classical LSTM widely used in NLP applications and other sequential data analysis tasks. Like classical LSTM, QLSTM is designed to process data sequences and capture long-term dependencies between the sequence elements. However, instead of using classical gates to perform computations, QLSTM uses quantum gates, which can perform complex transformations on qubits.

Now, some researchers are trying to implement NLP applications by using QML to increase the performance of algorithms. Quantum deep learning [36] is a research area where quantum computing properties are used to train network networks. QNN can be implemented by using VQC, which are parameterized circuits. Recently, researchers have been trying to use QNN for NLP applications, which may speed up computation.

Table 4
Quantum algorithms.

Quantum machine learning algorithms	Speedup
HHL algorithm [30]	Exponential
Quantum support vector machine [37]	Exponential
Quantum principle component analysis [38]	Exponential
Quantum linear discriminant analysis [39]	Exponential
Quantum decision trees [40]	Polynomial
Quantum linear regression [41]	Exponential
Quantum Boltzmann machines [42]	Exponential
Quantum nearest neighbor algorithm [43]	Exponential
Quantum k-means clustering [44]	Quadratic
Quantum Bayesian network [45]	Quadratic

Quantum natural language processing (QNLP) [46] is another research field that tries to use quantum mechanics property for natural language applications. QNLP uses categorical quantum mechanics, which relate syntactic sentence structure to quantum states and grammar to entanglement. Lambeq [47] is a QNLP framework that tries to solve various NLP applications. Discopy is one of the tools of the lambeq framework, which performs low-level operation means converting a compositional word model into quantum circuits. A compositional model finds the meaning of sentences. This model requires a huge amount of classical computation.

Riordan et al. [48] have proposed a hybrid workflow that encodes, process, and decode sentence meaning into the quantum circuit. They have shown how to represent meanings and calculate the similarity between different meaning representations. When the corpus size is increased, the circuit depth will increase. So This model is only applicable to the small and simple corpus.

Shi et al. [49] proposed a quantum-based neural network for binary text classification. They used quantum-inspired complex-valued word embedding (ICWE). A word may contain different meanings and equivalents, like a quantum state may hold different values. They have proven the feasibility and effectiveness of applying NLP and can solve the problem of text information loss.

QNN can also be applied in NLP applications like POS tagging. The author [50] demonstrate a QLSTM-based POS tagger for POS tagging. It is a basic model for POS tagging where they have used only two sentences. They have compared classical LSTM and QLSTM and demonstrated how QLSTM can learn. They gave an intuition that QNN can apply in POS tagging. It is a basic model that cannot apply to a large dataset.

Pandey et al. [51] have used the QLSTM model for POS tagging on the Mizo, a low-resource language dataset. They have experimented with a Mizo dataset which contains 30 000 words and 47 POS tags. But their experimental result is not satisfactory. They have tried with different numbers of qubits and other local simulators. They have investigated that current quantum devices do not apply to large datasets.

We have analyzed that until now, no QNN model is applicable for POS tagging of large datasets. In this paper, we implement QLSTM for POS tagging code-mixed datasets. Our experiment result is described in the result section. We have shown a comparison between classical LSTM and QLSTM results.

5. Platform

5.1. Amazon-braket

Amazon Braket [52] is a web-based quantum computing service provided by Amazon Web Services (AWS). It offers tools and libraries for quantum computing, including simulators and access to quantum processors from various hardware providers. With Amazon Braket, users can run quantum algorithms and experiments using a variety of programming languages, such as Python, and a range of development tools.

Users can access pre-built quantum algorithms and examples from the Amazon Braket SDK. Amazon braket provides access to simulators and quantum hardware and offers tools to design quantum circuits, optimize, and test. Users can use these tools to design and simulate quantum circuits, optimize them on different quantum hardware, and analyze their performance.

Amazon braket gives access to different quantum simulators like state vector simulator (SV1), tensor-based simulator (TV1), and density-based simulator (DM1) as well as other kinds of quantum devices like gate-based quantum computers (Rigetti, Lucy, Oxford Quantum Circuits), Rydberg atom qubits (QuEra), quantum photonics (Xanadu).

The SV1 is a general-purpose on-demand quantum simulator used to simulate quantum circuits. It can simulate up to 34 qubits. The TN1 simulator is a high-performance, tensor network simulator which is used for QML applications. It supports up to 50 qubits and a circuit depth of up to 1000. It is best suitable for sparse circuits. Previously both simulators were not best suited for noisy output. Actual quantum hardware always gives noisy output. If we want to run any quantum algorithms for noisy output, use the DM1 simulator based on the density matrix. These simulators can run multiple quantum circuits simultaneously.

Amazon braket also offers a hybrid quantum–classical simulator, where a user can simulate a quantum circuit on a classical computer for a few steps and then complete the computation on an actual quantum computer. Such a hybrid approach reduces the resources required to run a quantum algorithm, making it easier and more cost-effective to implement quantum computing in real life.

5.2. PennyLane

PennyLane [53] is an open-source, cross-platform Python library for quantum computing. Its different programming paradigms can execute and train quantum programs on various backends. It connects other machine learning libraries like PyTorch, TensorFlow, Keras, and Numpy with quantum computing.

PennyLane provides a convenient interface for designing and implementing QML algorithms and optimizing them using classical optimization techniques. It also provides a platform for developing hybrid quantum–classical algorithms, combining quantum and classical computation. This allows the users to connect the strength of classical and quantum computing to solve complex problems more efficiently.

Moreover, it supports different tools for quantum circuit optimization, error correction, and compilation. It provides a comprehensive set of quantum machine learning algorithms, which includes quantum vector machines, quantum neural networks, and quantum principal component analysis.

PennyLane is developed by Xanadu, a Canadian quantum computing company, and has a rapidly growing community of users and contributors. It is available for free and open-source on GitHub under the Apache License 2.0. With the continuous evolution of quantum computing, PennyLane is expected to play a vital role in the research and applications of advanced quantum machine learning.

5.3. Qiskit

Qiskit [54] is an open-source Python library for quantum computing. It is developed by IBM. It gives access to tools, libraries, and APIs that enable developers and researchers to design, simulate, and execute quantum circuits and applications on actual quantum hardware or simulators.

Qiskit provides various pre-build quantum algorithms and examples to understand the user. It also provides an interface to access IBM quantum real devices. It offers a range of features and capabilities, including tools for quantum circuit design and visualization, quantum algorithm development, and simulation.

Under the Apache License 2.0, Qiskit is freely downloadable and open-source on GitHub, consisting of a huge and active community of users and developers. Qiskit is widely used for research and development in quantum computing in academia and industry.

5.4. TensorFlow quantum

TensorFlow Quantum [55] is a software developed by Google for building QML algorithms framework. It combines quantum computing with TensorFlow, which is a popular machine-learning library for enabling the training of hybrid quantum–classical models of QML.

TensorFlow is written in Python and C++, and it provides a high-level API for defining and running computational graphs, allowing developers to specify complex models and algorithms easily. It also supports distributed computing and GPU acceleration, enabling faster training and inference of large-scale models. The range of quantum computing platforms provided by the library can also be used to simulate quantum circuits, including simulators for qubits and continuous variables.

It also can interface with the existing quantum computing hardware. This allows the developers to run quantum circuits on various platforms, including Google's and third-party quantum computing platforms. It is broadly applied in academia and industry for multiple applications like machine learning models, NLP applications, and recommendation systems. It is freely available open-source on GitHub under the Apache License 2.0.

6. Dataset description

Code-mixed [56] datasets are collections of text or speech data containing a mixture of two or more languages or language varieties within the same discourse or communication. Such datasets are generally used in NLP tasks, such as language identification [57], machine translation, named entity recognition, text summarization [58], sentiment analysis, and speech recognition, among others.

Meanwhile, code-mixing is a common phenomenon in multilingual societies, where people frequently switch between languages or mix them together in their communication. For example, a speaker may use words or phrases from different languages in the same sentence or use one language for some parts of a conversation and another for others. These code-mixed data are primarily seen in social media, where people tend to mix languages while communicating, discussing, and reviewing.

Table 5
Tag set of code mixed.

Category	Description
G_N	Noun
G_PRP	Pronoun
G_V	Verb
G_J	Adjective
G_R	Adverb
G_PRP	Demonstrative
G_SYM	Quantifier
G_PRT	Particles
CC	Conjunction
PSP	Pre- & Postposition
&	Numeral
DT	Determiner
G_X	Residual & Twitter-Specific

The dataset used in our experiment has been retrieved from ICON 2016 Code-Mixed data on Indian languages.¹ The dataset contains social media datasets in three different Indian language pairs: Bengali–English, Hindi–English, and Telugu–English of Facebook, WhatsApp, and Twitter social media data.

It is further classified into fine-grained code mixed data and coarse-grained code mixed data. Coarse-grained data consists of tagsets that describe the category of a particular word. There are 13 categories of tagsets defined for coarse-grained data, whereas in fine-grained data, each category is further divided into different types. An example of coarse-grained data and fine-grained data is mentioned below. For example, all noun types are tagged as G_N in coarse-grained datasets. On the other hand, a noun is further classified as a common noun, proper noun, verbal noun, and spatio-temporal noun, tagged as N_NN, N_NNP, N_NNV, and N_NST, respectively, in fine-grained datasets.

We have used a coarse-grained dataset in our experiment. Our system is in the early stage of development and hence, not highly effective; we have incorporated coarse-grained datasets for training our model. Fine-grained datasets consist of numerous tagsets, which could make it difficult to train our model and increase its complexity. The tag set of the same is briefly shown in Table 5. The Hindi–English dataset of sources from Facebook, Twitter, and Whatsapp consists of 1401, 1264, and 1264 sentences, respectively. For the Bengali–English dataset, the number of sentences is 782, 464, and 464 for Facebook, Twitter, and Whatsapp, respectively. The number of sentences in the Telugu–English dataset from sources Facebook, Twitter, and Whatsapp is 1280, 998, and 1228 respectively, as shown in Table 6.

Below shown is the example of the dataset before the preprocessing.

```
@bionicsix1  univ
@ @phanerozoic11 univ
@ @pari_cious univ
@ bohut hi QT_QTF
achay hi JJ
ayay hi V_VM
. univ RD_SYM
Mixed en JJ
dabay hi N_NN
Wala hi RP_RPD
mix en N_NN
n en CC
maida hi N_NN
. univ RD_SYM
Apna hi PR_PRL
hee hi RP_RPD
koi hi DM_DMI
taste en JJ
bana hi V_VM
liya hi V_VAUX
:) univ E
```

¹ <http://amitavadas.com/Code-Mixing.html>.

Table 6
Statistics of dataset.

Source	Dataset name	No. of sentences	No. of words
Facebook	Hindi–English	1401	14 651
	Bengali–English	782	7243
	Telugu–English	1280	7510
Whatsapp	Hindi–English	1264	9500
	Bengali–English	464	3482
	Telugu–English	998	6499
Twitter	Hindi–English	1264	9590
	Bengali–English	464	3370
	Telugu–English	1228	9572

7. Experiment

We are taking leverage quantum computing in NLP applications. It is the beginning phase of building QML algorithms for NLP applications. Here, We apply classical LSTM and QLSTM models, compatible with POS tagging of code mixed datasets. Our QLSTM model is inspired by QLSTM architecture [50].

7.1. System architecture

Long short-term memory (LSTM) [59] is a variant of a recurrent neural network (RNN) [60], which is used to handle longer dependencies in data. An RNN model is a neural network model used to process sequential data. But It suffers from a vanishing gradient problem when more data dependencies are encountered. The vanishing gradient problem occurs when the gradients used to update the network's weights become too small to be effective, making it difficult for the network to learn long-term dependencies. LSTM can solve this problem and is widely used in NLP applications.

The main difference between LSTM and RNN is that LSTM uses a memory cell to store information for an extended period. This memory cell's workflow depends on four gates: an input gate, a forget gate, an update gate, and an output gate. The input gate identifies how much new information is accepted to insert into the memory cell, and the forget gate chooses what information will be rejected. The data is updated based on the update gate. The output gate controls how much of the information stored in the memory cell is used to generate the output. Using these gates enables LSTM to selectively remember and forget information over time to handle long-term dependencies in the dataset. The below equations show the flow of information in LSTM.

$$i_t = \sigma(W_i[h_{t-1}, X_t] + b_i) \quad (10)$$

$$f_t = \sigma(W_f[h_{t-1}, X_t] + b_f) \quad (11)$$

$$o_t = \sigma(W_o[h_{t-1}, X_t] + b_o) \quad (12)$$

$$S_t = \tanh(W_s[h_{t-1}, X_t] + b_c) \quad (13)$$

$$h_t = o_t * \tanh(S_t) \quad (14)$$

where σ is the sigmoid function, i_t represents the workflow of the input gate, f_t represents the workflow of forget gate, o_t represents the workflow of the output gate, and S_t represents the workflow of the update gate. W_i , W_f , W_o , and W_s denotes the parameters of corresponding gate. $[h_{t-1}, X_t]$ is combination of hidden state and input data. b_n is corresponding bias of W_n , where n is type of gate.

LSTM is built by stacking memory cells and hidden cells. Fig. 2 shows a diagram of an LSTM, where X_t represents input at time instance t and h_{t-1} represents the previous hidden state. This model first takes words as input, combining the current input instance X_t and h_{t-1} , then performs word embedding. Then embedded word is processed by an LSTM layer, producing hidden states at each time instance. The final layer of this model maps the generated hidden states into tag space, which will be the tag of the corresponding time instance input X_t . In this model, stochastic gradient descent [61] is used as an optimizer that is best suitable for small datasets, and cross-entropy is used as a loss function.

7.2. Quantum long-short term memory

QLSTM is a quantum-based LSTM where VQC represents each gate of LSTM. It is a hybrid quantum–classical model, a combination of classical and quantum. Each VQC has three layers: data encoding layer, variational layer, and measurement layer (see Fig. 3).

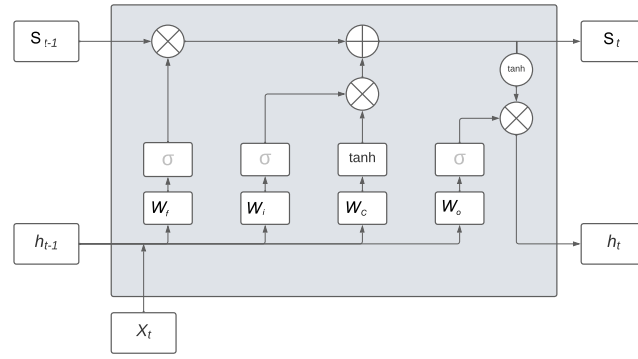


Fig. 2. A diagram of Long short-term memory.

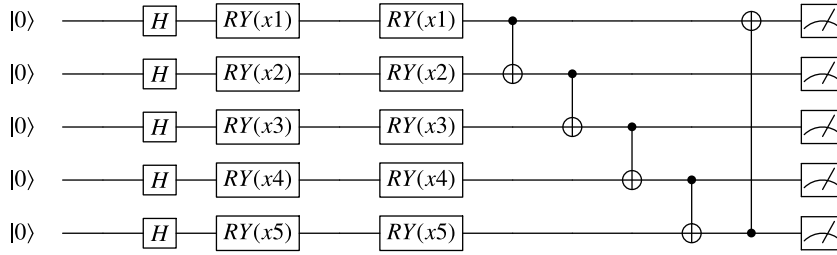


Fig. 3. A quantum circuit.

First, the data encoding layer converts classical data into quantum states. Here we use angle embedding, an encoding technique to encode the classical data into quantum states. The initial state of the quantum system will be $|0\rangle^N$, where N is the number of qubits. The first encoding step is converting the initial state into complex states using the Hadamard gate. The below-given equation explains how to convert initial states into complex states.

$$\begin{aligned}
 (H|0\rangle)^{\otimes N} &= \frac{1}{\sqrt{2^N}}(|0\rangle + |1\rangle)^{\otimes N} \\
 &= \frac{1}{\sqrt{2^N}}(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes \cdots \otimes (|0\rangle + |1\rangle) \\
 &= \frac{1}{\sqrt{2}} \sum_{i=0}^{2^N-1} |i\rangle
 \end{aligned} \tag{15}$$

Now, the RY gate is used to rotate the data corresponding y-axis. For example input data $X = (x_1, x_2, \dots, x_n)$ converted into quantum state by $RY(x_1), RY(x_2), \dots, RY(x_n)$. The proposed model takes words as input and combines the input instance with hidden states. Then use a word embedding technique to convert each input converted into word embedded vector, which dimension will be the number of qubits. Now, at a time, one input instance will be encoded.

Next, the Variational layer is used to apply quantum operations to process the encoded classical data. This layer consists of CNOT gates, single qubit rotation gates, and tunable parameters to train the QLSTM. At the end of the variational layer, there exists a measurement layer that measures the output of each circuit. Here we consider the expectation values of every qubit by measuring on a computational basis with quantum simulation software such as PennyLane.

All the blocks, like forget gate, input gate, update gate, and output gate of QLSTM, are stacked. First, QLSTM decides which information should pass through the cell state. This decision is made by forget gate. In Fig. 4, VQC1 represents forget gate. Next, QLSTM decides which information will be added to the cell state. The input layer determines this; in Fig. 4, VQC2 describes it. The update gate is used to construct new candidate values to store the cell value. In Fig. 4, it is represented by VQC3. Finally, the output gate will decide what output is based on cell state values, and VQC4 represents it.

This model is a hybrid classical-quantum model, where the quantum part processes the data, and the classical part is used to optimize the parameters. In this model, stochastic gradient descent [61] is used as an optimizer that is best suitable for small datasets, and cross-entropy is used as a loss function. The measurement layer is used for the observation of output. In this model, we use the Pauli-z operator's expectation value. The below equation explains this expectation used.

$$\langle \psi | \sigma_z | \psi \rangle = \langle 0 | U_0(x)^\dagger U_i(\alpha)^\dagger \sigma_z U_0(x) U_i(\alpha) | 0 \rangle \tag{16}$$

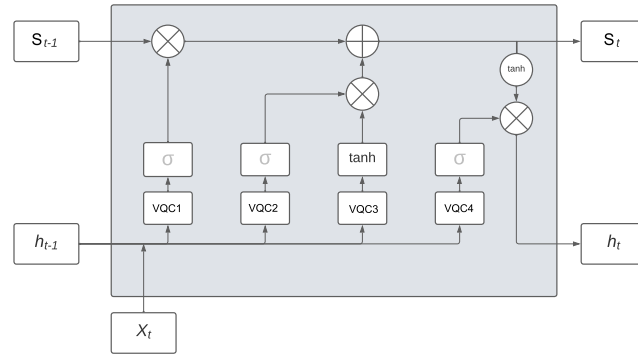


Fig. 4. An diagram of a Quantum long short term memory.

$$\alpha^{t+1} = \alpha^t - \mu \delta L(\alpha^t) \quad (17)$$

Algorithm 1 Proposed algorithm for QLSTM

Input: \mathbf{x}_t - input at time t

Input: \mathbf{h}_{t-1} - previous cell output at time $t - 1$

Input: \mathbf{c}_{t-1} - previous cell state at time $t - 1$

Input: $\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_o, \mathbf{W}_c$ - weight matrices for forget, input, output, and cell gates

procedure ENCODE(\mathbf{X})

for i to n_qubit **do**

 Hadamard(i)

 RY(x_i)

end for

end procedure

procedure VQC(\mathbf{X}, α)

for i to n_qubit **do**

 CNOT($i, i + 1$)

 RY(α_i)

end for

end procedure

procedure QLSTM(INPUT, N_QUBIT, EMBEDDING_DIM, HIDDEN_LAYER)

$\mathbf{f}_t = \sigma(\text{VQC}(\mathbf{W}_f, \text{Encode}(\mathbf{x}_t, \mathbf{h}_{t-1})))$

$\mathbf{i}_t = \sigma(\text{VQC}(\mathbf{W}_i, \text{Encode}(\mathbf{x}_t, \mathbf{h}_{t-1})))$

$\mathbf{o}_t = \sigma(\text{VQC}(\mathbf{W}_o, \text{Encode}(\mathbf{x}_t, \mathbf{h}_{t-1})))$

$\mathbf{c}_t = \tanh(\text{VQC}(\mathbf{W}_c, \text{Encode}(\mathbf{x}_t, \mathbf{h}_{t-1})))$

$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$

$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$

Output: \mathbf{h}_t - current cell output at time t

Output: \mathbf{c}_t - current cell state at time t

end procedure

▷ forget gate

▷ input gate

▷ output gate

▷ candidate cell state

▷ cell state

▷ cell output

7.3. Data preprocessing

Data preprocessing is a must to make our dataset suitable for our system. It not only ensures consistency but also helps improve the accuracy of the results and makes them more reliable.

The three major tasks involved in data preprocessing are cleaning, integration, and reduction. The dataset obtained from social media texts consists of texts, emojis, and emoticons. For dataset cleaning, the emojis and emoticons were removed. The misspelled words like typos, unreadable and unrecognized words were also removed for improving performance of our model, termed as dataset reduction in machine learning. Furthermore, for data integration, individual tokens or words were combined and integrated to form sentences. Example of the dataset after preprocessing is shown below:

Table 7
Different batches of datasets.

Source	Language	Batch 1	Batch 2	Batch 3	Batch 4	Batch 5	Batch 6
Facebook	Hindi	1–220	221–440	441–660	661–880	881–1100	1101–1320
	Bengali	1–220	221–440	441–660			
	Telugu	1–220	221–440	441–660	661–880	881–1100	
WhatsApp	Hindi	1–220	221–440	441–660	661–880	881–1100	
	Bengali	1–220	221–440				
	Telugu	1–220	221–440	441–660	661–880		
Twitter	Hindi	1–220	221–440	441–660	661–880	881–1100	
	Bengali	1–220	221–440				
	Telugu	1–220	221–440	441–660	661–880	881–1100	

Table 8
Parameters of classical LSTM.

No. of epoch	100
No. of layers	1
Size of word embedding	It depends on the number of words in training data
Size of hidden dimension	16
Learning rate	0.05

```

bohut : G_SYM
achay : G_J
ayay : G_V
. : G_X
Mixed : G_J
dabay : G_N
Wala : G_PRT
mix : G_N
n : CC
maida : G_N
. : G_X
Apna : G_PRP
hee : G_PRT
koi : G_PRP
taste : G_J
bana : G_V
liya : G_V

```

For the experiment, we are splitting the dataset into different batches. Where batch size is fixed (220 sentences), the number of batches differs for the different datasets. For example, Facebook-Hindi has six batches, and Facebook-Bengali has only three batches (see [Table 7](#)).

8. Experiment results

We perform the POS tagging of code mixed dataset by classical LSTM and QLSTM and compare the results of these two experiments. In these experiments, we use nine code-mixed datasets and split the datasets into training and testing. The dataset has three different social media texts: Whatsapp, Facebook, and Twitter, and each social media text is further divided into three different mixed languages: Hindi–English, Bengali–English, and Telugu–English.

8.1. Classical experiment

Here, we are using classical LSTM for POS tagging of code-mixed datasets. The dataset is divided into multiple batches, and each batch size is 220 sentences, exclusive of the test set. The size of each test set used is 22 sentences. [Table 8](#) shows the parameters of classical LSTM. For this experiment, cross-entropy is used as the loss function, and stochastic gradient descent is used as the optimizer algorithm to optimize the parameters of our model. The results of the experiment are given in [Table 9](#). From the results, we have observed that the accuracy for the Bengali dataset is the highest among all three languages, followed by Hindi and Telugu.

Table 9
Results of LSTM.

Source	Language	Batch 1	Batch 2	Batch 3	Batch 4	Batch 5	Batch 6	Average
		Accuracy	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy
Facebook	Hindi	55.26	38.96	33.8	38.02	34.74	30.98	38.62
	Bengali	55.45	53.18	61.36				56.67
	Telugu	50.47	42.81	44.08	47.63	49.52		46.9
WhatsApp	Hindi	54.45	50.78	48.16	56.02	57.06		53.29
	Bengali	62.75	60.68					61.77
	Telugu	38.8	34.46	33.49	42.71			37.37
Twitter	Hindi	54.45	50.26	49.21	32.98	51.83		47.75
	Bengali	45.5	47.5					46.5
	Telugu	56.45	40.19	47.24	45.45	49.28		47.32

Table 10
Parameters of QLSTM experiment.

No. of epoch	100
No. of qubits	5
No. of quantum layer	1
Size of word embedding	It depends on the number of words in training data
Size of hidden dimension	10
Learning rate	0.05

Table 11
Results of QLSTM.

Source	Language	Batch 1	Batch 2	Batch 3	Batch 4	Batch 5	Batch 6	Average
		Accuracy	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy
Facebook	Hindi	40.24	61.35	54.27	69.53	57.81	58.2	56.9
	Bengali	53.00	49.05	54.25				52.1
	Telugu	56.35	50.13	53.37	50.46	55.14		53.09
WhatsApp	Hindi	52.25	56.36	54.24	36.56	50.22		49.93
	Bengali	59.45	59.45					59.45
	Telugu	57.59	58.24	59.65	47.16			55.66
Twitter	Hindi	53.14	55.42	47.42	42.28	51.42		49.94
	Bengali	51.26	59.23					55.25
	Telugu	57.59	38.66	44.66	38.67	47.33		45.38

8.2. Quantum experiment

PennyLane integrated with the Pytorch library is a platform for quantum simulation in the QLSTM experiments. This QLSTM model is a hybrid classical-quantum model, where the quantum part processes the data, and the classical part is used to optimize the parameters. Stochastic gradient descent is used as an optimizer suitable for small datasets, and cross-entropy is used as a loss function.

Parameter tuning is a way to determine the best experiment result for the given dataset. So, In this experiment, different sets of parameters are tried to get the best result of the QLSTM POS tagger. But we get the best outcome for the given parameters in Table 10. Our experiment's result is shown in Table 11, which shows that QLSTM can learn from less data than classical LSTM. For example, the QLSTM's experiment result is higher than the classical LSTM's result for the Hindi–English of the Facebook dataset.

Since the number of sentences in each language dataset varies, the number of batches for each also varies between 2 to 6. For almost all batches, the accuracy is between 45% to 60%. The best accuracy is shown by the Bengali dataset, especially in Whatsapp.

9. Error analysis

9.1. Number of qubits

We found that by increasing the number of qubits from 1 to 5, the accuracy increased from 15% to 55%. This was due to an effect called underfitting, but after crossing the hurdle, there is a decrease in accuracy, possibly due to added noise or overfitting of the model. For example, when using ten qubits instead of 5, the accuracy may decrease from 55% to 40%. An example is the dataset curated from Facebook containing the Telugu–English pair is given in Table 12.

First

Table 12

Observation on number of qubits.

Sentence	Enduku	trend	avthundo	evariki	telidu
Actual labels	G_PRP	G_N	G_V	G_SYM	G_V
Predicted labels (5 qubits)	G_PRP	G_N	G_X	G_SYM	G_V
Predicted labels (10 qubits)	G_PRP	PSP	G_V	G_N	G_N

Table 13

Observation on number of epochs.

Sentence	ki	movie	release	hoeche	?
Actual labels	G_PRP	G_N	G_N	G_V	G_X
Predicted labels (100 epochs)	G_PRP	PSP	G_N	G_N	G_X
Predicted labels (200 epochs)	G_N	PSP	GH_N	G_N	G_N

Table 14

Observation on number of hidden dimension.

Sentence	ki	movie	release	hoeche	?
Actual labels	G_N	G_N	G_N	G_V	G_X
Predicted labels (Embedding dim. = 08)	G_PRP	PSP	G_N	G_N	G_X
Predicted (Embedding dim. = 15)	G_N	PSP	G_N	G_N	G_N
Predicted labels (Embedding dim. = 11)	G_PRP	GN	G_N	G_N	G_X

9.2. Number of epochs

We found that increasing the number of epochs to 100 increased the accuracy of our model. Still, on trying to increase epochs above 100, the accuracy decreased drastically, which may be due to the overfitting of the model to the training data as discussed in 9.1. For example, when using 200 epochs instead of 100, the accuracy decreased from 55% to 30%. An example is the data curated from Whatsapp containing the Bengali–English pair given in Table 13.

9.3. Hyper-parameter tuning

It was found that using a random search algorithm, the number of embedding layers should equal the 0.275th power of the number of parameters to achieve the best results. For example, if the model has 1000 parameters, the optimal number of embedding layers would be 2 (i.e., $1000^{0.275} = 2.04$). This hyper-parameter tuning approach ensures that the model is neither too complex nor too simple, which can lead to overfitting or underfitting. Several experiments were run with several embedding layers more than the calculated value, less than the estimated value, or exactly the computed value. The result of these experiments is shown in Table 14.

10. Conclusion

Quantum computing and machine learning is a rapidly evolving field of computer science. QML is a combination of quantum computing and machine learning. We can derive the benefit of QML algorithms in NLP applications. This paper has discussed the background of quantum computing and the basic idea of QML. We also provide POS tagging of social media code-mixed language by using classical LSTM and QLSTM. From our experiment, we have observed that QLSTM is better than classical LSTM for Facebook (Hindi–English, Telugu–English), WhatsApp (Telugu–English), and Twitter (Hindi–English, Bengali–English).

Our future work consists of building QML algorithms, which will be efficient for NLP applications. We will use an efficient encoding method, which will be able to convert classical data into quantum states and build a robust VQC for ML models that can be trainable for large datasets.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

Acknowledgments

The work presented here falls under the Research Project Grant Ref. No. N-21/17/2020-NeGD supported by MeitY Quantum Computing Applications Lab (QCAL), India.

References

- [1] L. Gyongyosi, S. Imre, A survey on quantum computing technology, *Comp. Sci. Rev.* 31 (2019) 51–71.
- [2] J.D. Hiday, *Quantum Computing: An Applied Approach*, Springer, New York, NY, 2019, pp. xiv–324, <http://dx.doi.org/10.1007/978-3-030-21735-4>.
- [3] B. Wang, F. Hu, H. Yao, C. Wang, Prime factorization algorithm based on parameter optimization of ising model, *Sci. Rep.* 10 (1) (2020) 1–10.
- [4] P. Kwiat, J. Mitchell, P. Schwindt, A. White, Grover's search algorithm: an optical approach, *J. Modern Opt.* 47 (2–3) (2000) 257–266.
- [5] R. Jozsa, N. Linden, On the role of entanglement in quantum-computational speed-up, *Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci.* 459 (2036) (2003) 2011–2032.
- [6] A.Y. Shiekh, The role of quantum interference in quantum computing, *Internat. J. Theoret. Phys.* 45 (9) (2006) 1646–1648.
- [7] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, S. Lloyd, Quantum machine learning, *Nature* 549 (7671) (2017) 195–202.
- [8] S. Russell, S. Russell, P. Norvig, E. Davis, *Artificial Intelligence: A Modern Approach*, in: Prentice Hall Series in Artificial Intelligence, Prentice Hall, 2010, pp. XIX–79.
- [9] D.W. Otter, J.R. Medina, J.K. Kalita, A survey of the usages of deep learning for natural language processing, *IEEE Trans. Neural Netw. Learn. Syst.* 32 (2) (2020) 604–624.
- [10] A. Caggiano, J. Zhang, V. Alfieri, F. Caiazzo, R. Gao, R. Teti, Machine learning-based image processing for on-line defect recognition in additive manufacturing, *CIRP Ann.* 68 (1) (2019) 451–454.
- [11] A.I. Khan, S. Al-Habsi, Machine learning in computer vision, *Procedia Comput. Sci.* 167 (2020) 1444–1451.
- [12] T.M. Khan, A. Robles-Kelly, Machine learning: Quantum vs classical, *IEEE Access* 8 (2020) 219275–219294.
- [13] F. Phillipson, Quantum machine learning: Benefits and practical examples, in: *QANSWER*, 2020, pp. 51–56.
- [14] F. Stahlberg, Neural machine translation: A review, *J. Artificial Intelligence Res.* 69 (2020) 343–418.
- [15] S. Adhikari, et al., Nlp based machine learning approaches for text summarization, in: *2020 Fourth International Conference on Computing Methodologies and Communication, ICCMC, IEEE*, 2020, pp. 535–538.
- [16] M. Kobayashi, K. Takeda, Information retrieval on the web, *ACM Comput. Surv. (CSUR)* 32 (2) (2000) 144–173.
- [17] M.A.C. Soares, F.S. Parreiras, A literature review on question answering techniques, paradigms and systems, *J. King Saud Univ. Comput. Inf. Sci.* 32 (6) (2020) 635–646.
- [18] D. Jurafsky, *Speech & Language Processing*, Pearson Education India, 2000.
- [19] J. Awwalu, S.E.-Y. Abdullahi, A.E. Ewviekpaefe, Parts of speech tagging: a review of techniques, *Fudma J. Sci.* 4 (2) (2020) 712–721.
- [20] P. Pakray, G. Majumder, A. Pathak, An HMM based pos tagger for POS tagging of code-mixed Indian social media text, in: *Social Transformation–Digital Way: 52nd Annual Convention of the Computer Society of India, CSI 2017, Kolkata, India, January 19–21, 2018, Revised Selected Papers 52*, Springer, 2018, pp. 495–504.
- [21] A. Chiche, B. Yitagesu, Part of speech tagging: a systematic review of deep learning and machine learning approaches, *J. Big Data* 9 (1) (2022) 1–25.
- [22] J.-L. Brylinski, R. Brylinski, Universal quantum gates, in: *Mathematics of Quantum Computation*, Chapman and Hall/CRC, 2002, pp. 117–134.
- [23] S. Sim, P.D. Johnson, A. Aspuru-Guzik, Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms, *Adv. Quant. Technol.* 2 (12) (2019) 1900070.
- [24] R. LaRose, B. Coyle, Robust data encodings for quantum classifiers, *Phys. Rev. A* 102 (3) (2020) 032420.
- [25] M. Weigold, J. Barzen, F. Leymann, M. Salm, Data encoding patterns for quantum computing, in: *Proceedings of the 27th Conference on Pattern Languages of Programs*, 2020, pp. 1–11.
- [26] M. Weigold, J. Barzen, F. Leymann, M. Salm, Encoding patterns for quantum algorithms, *IET Quant. Commun.* 2 (4) (2021) 141–152.
- [27] S.Y.-C. Chen, C.-H.H. Yang, J. Qi, P.-Y. Chen, X. Ma, H.-S. Goan, Variational quantum circuits for deep reinforcement learning, *IEEE Access* 8 (2020) 141007–141024.
- [28] K. Mitarai, M. Negoro, M. Kitagawa, K. Fujii, Quantum circuit learning, *Phys. Rev. A* 98 (3) (2018) 032309.
- [29] A. Macaluso, L. Clissa, S. Lodi, C. Sartori, A variational algorithm for quantum neural networks, in: *Computational Science–ICCS 2020: 20th International Conference, Amsterdam, the Netherlands, June 3–5, 2020, Proceedings, Part VI 20*, Springer, 2020, pp. 591–604.
- [30] A.W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations, *Phys. Rev. Lett.* 103 (15) (2009) 150502.
- [31] K. Najafi, S. Yelin, X. Gao, The development of quantum machine learning, *Harv. Data Sci. Rev.* (2022) 1–19, <http://dx.doi.org/10.1162/99608f92.5a9fd72c>.
- [32] A. Callison, N. Chancellor, Hybrid quantum-classical algorithms in the noisy intermediate-scale quantum era and beyond, *Phys. Rev. A* 106 (1) (2022) 010101.
- [33] J. Bausch, Recurrent quantum neural networks, *Adv. Neural Inf. Process. Syst.* 33 (2020) 1368–1379.
- [34] I. Cong, S. Choi, M.D. Lukin, Quantum convolutional neural networks, *Nat. Phys.* 15 (12) (2019) 1273–1278.
- [35] S. Chen S. Y. C., Y.L.L. Fang, Quantum long short-term memory, 2022, pp. 8622–8626.
- [36] S. Garg, G. Ramakrishnan, Advances in quantum deep learning: An overview, 2020, pp. 1–17, arXiv preprint [arXiv:2005.04316](https://arxiv.org/abs/2005.04316).
- [37] P. Rebentrost, M. Mohseni, S. Lloyd, Quantum support vector machine for big data classification, *Phys. Rev. Lett.* 113 (13) (2014) 130503.
- [38] S. Lloyd, M. Mohseni, P. Rebentrost, Quantum principal component analysis, *Nat. Phys.* 10 (9) (2014) 631–633.
- [39] I. Cong, L. Duan, Quantum discriminant analysis for dimensionality reduction and classification, *New J. Phys.* 18 (7) (2016) 073011.
- [40] E. Farhi, S. Gutmann, Quantum computation and decision trees, *Phys. Rev. A* 58 (2) (1998) 915.
- [41] G. Wang, Quantum algorithm for linear regression, *Phys. Rev. A* 96 (1) (2017) 012335.
- [42] M.H. Amin, E. Andriyash, J. Rolfe, B. Kulchitsky, R. Melko, Quantum Boltzmann machine, *Phys. Rev. X* 8 (2) (2018) 021050.
- [43] N. Wiebe, A. Kapoor, K.M. Svore, Quantum nearest-neighbor algorithms for machine learning, *Quant. Inf. Comput.* 15 (3–4) (2015) 318–358.
- [44] C. Gong, Z. Dong, A. Gani, H. Qi, Quantum k-means algorithm based on trusted server in quantum cloud computing, *Quantum Inf. Process.* 20 (2021) 1–22.
- [45] G.H. Low, T.J. Yoder, I.L. Chuang, Quantum inference on Bayesian networks, *Phys. Rev. A* 89 (6) (2014) 062315.
- [46] B. Coecke, G. de Felice, K. Meichanetzidis, A. Toumi, Foundations for near-term quantum natural language processing, 2020, arXiv preprint [arXiv:2012.03755](https://arxiv.org/abs/2012.03755).
- [47] D. Kartsaklis, I. Fan, R. Yeung, A. Pearson, R. Lorenz, A. Toumi, G. de Felice, K. Meichanetzidis, S. Clark, B. Coecke, lambeq: An efficient high-level python library for quantum NLP, 2021, pp. 1–19, arXiv preprint [arXiv:2110.04236](https://arxiv.org/abs/2110.04236).

- [48] L.J. O’Riordan, M. Doyle, F. Baruffa, V. Kannan, A hybrid classical-quantum workflow for natural language processing, *Mach. Learn.: Sci. Technol.* 2 (1) (2020) 015011.
- [49] J. Shi, Z. Li, W. Lai, F. Li, R. Shi, Y. Feng, S. Zhang, Two end-to-end quantum-inspired deep neural networks for text classification, *IEEE Trans. Knowl. Data Eng.* 35 (4) (2023) 4335–4345, <http://dx.doi.org/10.1109/TKDE.2021.3130598>.
- [50] R. Di Sipio, J.-H. Huang, S.Y.-C. Chen, S. Mangini, M. Worring, The dawn of quantum natural language processing, in: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2022*, pp. 8612–8616, <http://dx.doi.org/10.1109/ICASSP43922.2022.9747675>.
- [51] S. Pandey, P. Dadure, M.V. Nunsanga, P. Pakray, Parts of speech tagging towards classical to quantum computing, in: *2022 IEEE Silchar Subsection Conference, SILCON, IEEE, 2022*, pp. 1–6.
- [52] C. Gonzalez, Cloud based qc with amazon braket, *Digit. Welt* 5 (2021) 14–17.
- [53] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M.S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, et al., PennyLane: Automatic differentiation of hybrid quantum-classical computations, 2018, arXiv preprint [arXiv:1811.04968](https://arxiv.org/abs/1811.04968).
- [54] A. Cross, The IBM q experience and QISKit open-source quantum computing software, in: *APS March Meeting Abstracts, Vol. 2018, 2018*, pp. L58–003.
- [55] M. Broughton, G. Verdon, T. McCourt, A.J. Martinez, J.H. Yoo, S.V. Isakov, P. Massey, R. Halavati, M.Y. Niu, A. Zlokapa, et al., Tensorflow quantum: A software framework for quantum machine learning, 2020, arXiv preprint [arXiv:2003.02989](https://arxiv.org/abs/2003.02989).
- [56] A. Jamatia, B. Gambäck, A. Das, Part-of-speech tagging for code-mixed english-hindi twitter and facebook chat messages, in: *Proceedings of the International Conference Recent Advances in Natural Language Processing, 2015*, pp. 239–248.
- [57] E. Ambikairajah, H. Li, L. Wang, B. Yin, V. Sethu, Language identification: A tutorial, *IEEE Circuits Syst. Mag.* 11 (2) (2011) 82–108.
- [58] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E.D. Trippe, J.B. Gutierrez, K. Kochut, Text summarization techniques: a brief survey, 2017, arXiv preprint [arXiv:1707.02268](https://arxiv.org/abs/1707.02268).
- [59] V.C. Lobo Neto, L.A. Passos, J.P. Papa, Evolving long short-term memory networks, in: V.V. Krzhizhanovskaya, G. Závodszy, M.H. Lees, J.J. Dongarra, P.M.A. Sloat, S. Brissos, J. Teixeira (Eds.), *Computational Science – ICCS 2020, Springer International Publishing, Cham, 2020*, pp. 337–350.
- [60] M. Hibat-Allah, M. Ganahl, L.E. Hayward, R.G. Melko, J. Carrasquilla, Recurrent neural network wave functions, *Phys. Rev. Res.* 2 (2) (2020) 023358.
- [61] R. Sweke, F. Wilde, J. Meyer, M. Schuld, P.K. Faehrmann, B. Meynard-Piganeau, J. Eisert, Stochastic gradient descent for hybrid quantum-classical optimization, *Quantum* 4 (2020) 314, <http://dx.doi.org/10.22331/q-2020-08-31-314>.