Order N°: …
People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University of Mila
Institute of Mathematics and Computer Sciences
Department of Computer Science

Thesis presented to obtain the degree of
Doctorate in Computer Science
Third cycle doctoral thesis
Specialty: Networks and Distributed Systems

Presented by : Nour El Houda OUAMANE

Quantum Machine Learning Applied To Natural Language Processing

Members of the jury:

Pr. Nardjes BOUCHEMAL	University of Mila	President
Pr. Hacene BELHADEF	University of Constantine 2	Supervisor
Dr. Abdelhalim SAADI	University of Setif 1	Co-Supervisor
Dr. Madjed BENCHEIKHLEHOCINE	University of Mila	Examiner
Dr. Mounira BOUZAHZAH	University of Mila	Examiner
Dr. Mourad BOUZENADA	University of Constantine 2	Examiner

Academic Year 2025/2026

Acknowledgements

First and foremost, praises and thanks to Allah.

I would like to thank my dear husband, Pr. Imad Eddine Lakhdari, for his unwavering support and for belief in me. This thesis would not have been possible without his unconditional love, endless patience, and constant encouragement through every challenging phase of this work.

I would like to express my deep and sincere gratitude to my research supervisor, Pr. Hacene Belhadef, for giving me the opportunity to conduct this research and for providing invaluable guidance throughout its development. His dynamism, vision, sincerity, and motivation have deeply inspired me.

I extend sincere thanks to the members of the jury: Pr. Nardjes Bouchemal, from Mila university, Dr. Madjed Bencheikhlehocine from Mila university, Dr. Mounira Bouzahzah from Mila university, Dr. Mourad Bouzenada from Constantine 2 university and Dr. Abdelhalim SAADI, from Setif 1 university, for the time they spent reviewing this work, i am grateful for the attention they paid to my work.

I would like to thank the head of the Mathematics Department, Dr. Samira Boukaf, for her support during the completion of my PhD thesis.

I owe my deepest gratitude to my parents for their unconditional love and endless encouragement. I cannot forget to also thank my family and any person, near or far, who contributed to the achievement of this research work.

Finally, a very special dedication goes to my little children. Thank you for being the brightest light in my life; your innocent smiles, laughter, and hugs were the ultimate source of energy and motivation that kept me moving forward every single day.

Nour El Houda Ouamane

Résumé

Avec l'expansion continue des réseaux sociaux, les internautes partagent leurs pensées et points de vue via divers supports, comme les textes. Malgré cette participation active, l'analyse des sentiments demeure un défi majeur en raison du volume important de textes provenant de sources diverses. Elle a suscité un intérêt considérable dans le domaine de traitement automatique du langage naturel (NLP). Elle offre des informations précieuses sur l'opinion publique, les retours clients et les expériences utilisateurs. Par ailleurs, la mécanique quantique a transformé notre compréhension de l'environnement, et l'une de ses branches, l'apprentissage automatique quantique (QML), qui a démontré des résultats théoriques précis et cohérents. Cependant, les approches QML appliquées à l'analyse des sentiments restent encore au stade théorique. Dans cette thèse, nous nous concentrons sur l'évaluation d'une approche de QML, à savoir les réseaux de neurones convolutifs quantiques (QCNN), pour la classification des critiques de films. Ce modèle représente une conception de réseau de neurones quantiques inspirée des réseaux de neurones convolutifs (CNN) et utilisant exclusivement des circuits quantiques paramétrés. De plus, nous étudions comment différents modèles QCNN, distingués par le nombre de qubits utilisés et les structures des circuits quantiques paramétrés, peuvent améliorer les performances de la classification des sentiments sur des ensembles de données artificiels et réels de critiques de films.

Mots Clés. L'apprentissage automatique quantique, Le traitement automatique du langage naturel, Les réseaux de neurones convolutifs quantiques, L'analyse des sentiments.

Abstract

As social networking platforms continue to grow, users increasingly share their thoughts and opinions through various forms of text. Despite this active engagement, sentiment analysis remains a significant challenge due to the vast volume of textual data generated from diverse sources. This task has gained considerable attention within the field of Natural Language Processing (NLP), as it provides valuable insights into public opinion, user experiences, and customer feedback. Meanwhile, quantum mechanics has reshaped our understanding of the world, and one of its emerging branches Quantum Machine Learning (QML) has shown promising theoretical results. However, the application of QML in sentiment analysis is still limited and remains largely at an experimental or theoretical stage. In this thesis, we evaluate one such QML approach: Quantum Convolutional Neural Networks (QCNNs) for movie review classification. QCNNs are quantum neural network architectures inspired by classical Convolutional Neural Networks (CNNs), but implemented entirely using parameterized quantum circuits. We investigate how different QCNN configurations with different number of qubits and structure of their parameterized circuits affect sentiment classification performance on both artificial and real movie review datasets.

Key words. Natural language processing, Quantum machine learning, Sentiment analysis, Quantum convolutional neural networks, Quantum computing.

ملخص

مع استمرار توسع شبكات التواصل الاجتماعي، يتشارك مستخدمو الإنترنت أفكارهم ووجهات نظرهم يوميًا، مستخدمين وسائط متنوعة كالنصوص والصور ومقاطع الفيديو والخطاب. ومع ذلك، ورغم المشاركة النشطة، لا يزال تحليل المشاعر يُمثل تحديًا بالغ الأهمية نظرًا للكم الهائل من النصوص الواردة من مصادر متنوعة وأفراد ذوي عقليات مختلفة. وقد حظي تحليل المشاعر باهتمام كبير في مجال أبحاث فهم اللغة، إذ يُقدم رؤى قيّمة حول الرأي العام، وآراء العملاء، وتجارب المستخدمين.

من ناحية أخرى, لقد غيّرت ميكانيكا الكم طريقة فهمنا لبيئتنا، وقد أظهر أحد فروعها، وهو التعلم الآلي الكمي (QML)، نتائج نظرية دقيقة ومتسقة، إلا أن مناهج QML المستخدمة في تحليل المشاعر لا تزال غير كافية وفي مراحلها النظرية. في هذه الرسالة، نُركز على تقييم أحد مناهج QML، وهو شبكات QCNN في تصنيف مراجعات الأفلام. يُمثل هذا النموذج تصميمًا لشبكة عصبية كمومية مستوحى من الشبكة العصبية التلافيفية (CNN) ويستخدم حصريًا دوائر كمية مُعلمة. مع ذلك، يلزم تحويل بيانات النصوص الكلاسيكية إلى حالات كمية، ولذلك نستخدم ثلاث تقنيات ترميز كمي، وهي ترميز الزاوية، وترميز السعة، وترميز متعدد الحدود الكمي اللحظي (IQP). نبحث في كيفية تحسين أداء تصنيف تحليل المشاعر عبر مجموعات البيانات الاصطناعية والحقيقية لمراجعات الأفلام، من خلال نماذج QCNN المختلفة، التي تتميز بعدد البتات الكمومية المستخدمة وهياكل الدوائر الكمومية ذات المعلمات.

Publications of the author

I. Journal Papers:

1. **Ouamane, N. E. H**., Belhadef, H., Haddad, M. (2025). Sentiment analysis of movie reviews based on quantum convolutional neural networks. The Journal of Supercomputing, 81(16), 1-20. https://link.springer.com/article/10.1007/s11227-025-07965-6

2. **Ouamane, N. E. H**., Belhadef, H., Saadi, A. (2025). Comparative Study of Quantum and Classical Convolutional Neural Networks for Sentiment Analysis. Submitted.

II. Conference Paper:

Ouamane, N. E. H., Belhadef, H. (2022). Deep reinforcement learning applied to NLP: a brief survey. The 2nd International Conference on New Technologies of Information and Communication (NTIC) (pp. 1-5). IEEE. https://easychair.org/cfp/NTIC22

III. Chapter book:

Ouamane, N. E. H., Belhadef, H. (2023). Proposed Model for QCNN-Based Sentimental Short Sentences Classification. In International Conference of Reliable Information and Communication Technology (pp. 214-223). Cham: Springer Nature Switzerland. https://link.springer.com/chapter/10.1007/978-3-031-59707-7_19

IV. Workshop and Diploma:

Ouamane, N. E. H. (2022). Quantum Computing and Programming. QBRONE using QISKIT and QWORLD. QTurkey. Diploma Number: QBronze101-37.
https://qworld.net/qbronze83/

List of Figures

List of Tables

List of abbreviations

NLP	Natural Language Processing
NLU	Natural Language Understanding
NLG	Natural Language Generation
ML	Machine Learning
MT	Machine Translation
DL	Deep Learning
CNNs	Convolutional Neural Networks
RNNs	Recurrent Neural Networks
LSTM	Long Short Term Memory
GRU	Gated Recurrent Unit
BERT	Bidirectional Encoder Representations from Transformers
POS	Parts of Speech
NER	Named Entity Recognitions
SVMs	Support Vector Machines
NP	Noun Phrase
VP	Verb Phrase
NER	Named entity recognition
SRL	Semantic Role Labeling
BSLP	Blank Slate Language Processor
BSTM	Bayesian Sentence based Topic Model

FGB	Factorization with Given Bases
TAOS	Topic Aspect-Oriented Summarization
MeSH	Medical Subject Headings
MEDLEE	MEDical Language Extraction and Encoding System
QML	Quantum Machine Learning
QCNNs	Quantum Convolutional Neural Networks
LR	Logistic Regression
RF	Random Forest
DT	Decision Tree
AI	Artificial Intelligence
MERA	Multiscale Entanglement Renormalization Ansatz
MNIST	Modified National Institute of Standards and Technology
CBOW	Continuous Bag of Words
U-Gate	Unitary Gate
P-Gate	Phase
IQP	Instantaneous Quantum Polynomial-time

Contents

General Introduction

General background

Language is a highly structured system of symbols and grammatical rules used for communication and information exchange between individuals. Natural Language Processing (NLP) is an interdisciplinary field combining artificial intelligence, computer science, and linguistics to enable computational systems to understand, interpret, analyze, and generate human language in both written and spoken forms. Its primary objective is to bridge the communication gap between humans and machines, allowing computers to process natural language in a meaningful, contextual, and intelligent manner resembling human understanding.

As vast volumes of unstructured text and speech data continue to be generated daily across global digital platforms, NLP plays a critical role in transforming this information into structured representations that machines can analyze and utilize effectively. In recent decades, the paradigm of NLP has fundamentally evolved from rule-based approaches grounded in formal linguistic theory to data-driven statistical methods, and ultimately to powerful modern Machine Learning (ML) architectures.

Machine Learning, a core branch of artificial intelligence, focuses on designing algorithms capable of automatically extracting patterns from complex datasets to make predictions or decisions with minimal human intervention. The rapid advancement of ML has become the primary driving force behind the evolution of modern NLP. While traditional linguistic approaches provided valuable early insights, they frequently struggled with the structural complexity, syntactic variability, and semantic ambiguity inherent in human communication. The integration of ML introduced a major shift, allowing computational models to learn latent language patterns directly from corpora rather than

depending on manually crafted rules. These techniques—ranging from classical algorithms such as Naïve Bayes and Hidden Markov Models to state-of-the-art deep learning architectures—have significantly enhanced performance across a wide spectrum of NLP tasks, including sentiment analysis and text categorization. Modern NLP systems now form the critical infrastructure behind ubiquitous real-world applications, such as virtual assistants, search engines, social media monitoring platforms, and automated customer support systems.

Limitations of Machine Learning in NLP

Despite these impressive achievements, contemporary ML faces several fundamental constraints, including a lack of rigorous theoretical foundations, limited model interpretability ("black-box" architectures), and an escalating dependence on massive training datasets alongside substantial computational and environmental resources. Furthermore, NLP presents distinct challenges inherent to the mathematical nature of language data, such as handling long-tail distributions, the inability to natively process symbolic structures, and limited efficiency in complex logical inference tasks.

Linguistic data typically follows a power-law distribution, meaning that as a text corpus expands, the vocabulary size grows sub-linearly but continuously. Consequently, regardless of the scale of the training dataset, models will inevitably encounter out-of-vocabulary terms or rare semantic structures. Addressing this long-tail problem remains a critical challenge that classical ML struggles to resolve independently.

Moreover, human language is inherently symbolic, which differs fundamentally from the continuous, real-valued vector spaces leveraged by classical neural networks. In practice, symbolic language data must be mapped into dense vector representations (embeddings) to be processed by deep architectures, and the subsequent outputs are mapped back into text symbols. However, a vast reserve of NLP knowledge exists explicitly in symbolic form, including lexical databases (e.g., WordNet), formal linguistic rules (e.g., syntax grammars), and structured world knowledge (e.g., knowledge graphs). Current deep learning frameworks have not fully exploited this symbolic knowledge. While symbolic representations are easily interpretable and deterministic, vector representations

excel at handling ambiguity, semantic nuance, and statistical noise. Consequently, a major open question in contemporary NLP research is how to effectively integrate symbolic and vector representations to leverage the complementary strengths of both paradigms [97, 131].

The Emergence of Quantum Computation

> Quantum computation is a distinctively new way of harnessing nature, It will be the first technology that allows useful tasks to be performed in collaboration between parallel universes
>
> *David Deutsch*

Quantum computers exploit the fundamental principles of quantum mechanics—such as superposition, entanglement, and interference—to resolve computational problems that are practically intractable for classical architectures. Over the past four decades, quantum computing has transitioned from an abstract theoretical proposition into physical hardware capable of executing tasks beyond the reach of the most powerful classical supercomputers.

In 1980, Benioff [20] first translated the abstract concept of computation into a physical framework by designing a quantum mechanical system whose unitary time-evolution simulated the state transitions of a classical Turing machine. This established the foundational proof that quantum systems could natively simulate classical computation. Concurrently, Manin [108] approached the problem from the converse perspective, demonstrating that simulating a generic quantum system on a classical computer requires exponential time relative to the system size. Shortly thereafter, Feynman [60, 61] reached the same conclusion independently and proposed a disruptive solution: constructing hardware operating natively on quantum principles to simulate nature efficiently.

To formalize these ideas, Deutsch [46] introduced the universal Quantum Turing Machine and the quantum circuit model in 1985. Subsequently, in 1992, the Deutsch–Jozsa algorithm [47] provided the first mathematical demonstration that a quantum approach

could solve a specific problem exponentially faster than any deterministic classical algorithm.

In the contemporary era, demonstrating "quantum computational advantage"—the experimental verification that a quantum processor can execute a specific computational task faster than any known classical simulation—has become a central milestone. In late 2019, a 53-qubit superconducting quantum processor claimed this milestone by sampling random quantum circuits [10]. While this initial claim faced immediate optimization challenges from classical algorithms reducing the simulation time significantly [139, 200], subsequent experiments utilizing 76-photon linear optical systems [208] and 66-qubit superconducting processors [197, 209] established sampling benchmarks where equivalent classical simulations were estimated to require thousands of years. With quantum hardware now demonstrating the capability to surpass classical limits on specific tasks, a pivotal question emerges: can this quantum advantage be harnessed to solve practically useful, real-world problems?

The Rationale for Quantum Machine Learning in NLP

> The reason we say it is 'quantum native' is that language seems to want to live on quantum
>
> *Bob Coecke*

The convergence of quantum computing and machine learning has given rise to the rapidly expanding field of Quantum Machine Learning (QML). This discipline develops quantum analogs of classical machine learning algorithms to leverage the distinct mathematical structure of quantum states for superior computational efficiency. QML models offer the potential to extract complex statistical patterns from high-dimensional spaces using significantly fewer training examples and lower runtime bounds [137].

These distinct advantages render QML uniquely suited for the structural complexities of NLP. Quantum states natively encode exponentially large vector spaces via tensor products, aligning perfectly with the high-dimensional demands of modern linguistic rep-

resentations. Furthermore, the principle of quantum superposition allows a system to represent multiple contextual meanings and semantic ambiguities simultaneously, mirroring the natural fluid properties of human language. Beyond state representation, QML frameworks provide mathematical acceleration for core algebraic operations fundamental to NLP, such as similarity searches, high-dimensional clustering, and large-scale matrix vectorization.

Research problem and objectives

Among various emerging QML architectures, Quantum Convolutional Neural Networks (QCNNs) have emerged as a powerful paradigm. Drawing inspiration from classical convolutional networks, QCNNs utilize specialized quantum circuits characterized by alternating convolutional and pooling layers. This hierarchical design yields crucial advantages, including a lower circuit depths that reduce quantum decoherence, and a structure well-suited for deployment on near-term Noisy Intermediate-Scale Quantum (NISQ) hardware.

While classical CNNs require exceptionally deep architectures and massive training sets to capture local features within text sequences, QCNNs leverage quantum mechanics to extract subtle structural features and semantic patterns with enhanced sample efficiency. This capability is highly advantageous for text categorization and sentiment analysis, where identifying local keyword correlations (n-grams) is critical for accurate classification.

Motivated by these structural advantages, the primary objective of this thesis is to investigate, design, and implement a novel QCNN framework tailored specifically for NLP text categorization tasks, focusing on sentiment analysis. This work evaluates diverse quantum data encoding strategies—such as angle and amplitude encoding—alongside various topologies of Parameterized Quantum Circuits (PQCs) to optimize feature extraction performance on text datasets [133]. Through this contribution, we aim to establish a reliable, hardware-efficient pipeline for processing and classifying linguistic data on quantum architectures.

Thesis plan

The remainder of this thesis is organized into four primary chapters, structured as follows:

Chapter 1 provides a comprehensive overview of the fundamentals of Natural Language Processing. It defines the core linguistic components of NLP, outlines its historical chronology from rule-based systems to deep learning, and reviews the standard benchmarks and text classification tasks that define the field.

Chapter 2 establishes the theoretical foundations of quantum mechanics and quantum information theory. It details the general principles of quantum gates, qubits, and circuit compilation, and reviews the seminal quantum algorithms that demonstrated the first instances of computational advantage.

Chapter 3 introduces the theoretical framework of Quantum Machine Learning (QML). It outlines dominant QML model architectures, provides a comparative analysis of their mechanics against classical machine learning, and surveys the software tools, SDKs, and hardware platforms available in the current ecosystem. Crucially, this chapter details the mathematical and structural principles underpinning the application of QML models to natural language structures.

Chapter 4 presents the core empirical contribution of this research: the design, implementation, and evaluation of a QCNN classifier optimized for text categorization using a movie review sentiment dataset. We first review recent related literature at the intersection of QML and sentiment analysis. Next, we detail our proposed methodology, encompassing data preprocessing, classical vector embedding, and quantum circuit encoding. Finally, we present our experimental results, provide a comparative performance analysis against a classical CNN baseline, and conclude with an in-depth discussion of our observations, architectural insights, and remaining hardware limitations.

The thesis concludes with a summary of its primary contributions and outlines promising directions for future research in the field of Quantum machine learning in NLP.

Chapter 1

Fundamentals of Natural Language Processing

1.1 Introduction

Natural Language Processing (NLP) is a major field of artificial intelligence that focuses on enabling computers to understand, analyze, and generate human language. Over the years, NLP has evolved considerably, moving from simple rule-based approaches to advanced machine learning and deep learning techniques capable of handling complex linguistic tasks. This evolution has significantly expanded the capabilities and applications of intelligent language-processing systems in various domains.

This chapter presents the fundamental concepts and development of NLP. It first introduces the main components of NLP. The chapter then reviews the historical evolution of NLP, highlighting the major technological and methodological advancements that shaped the field over time. Finally, it explores several important applications of NLP, demonstrating its growing impact in areas such as machine translation, sentiment analysis, information extraction and summarization. Through these discussions, the chapter establishes the theoretical background necessary for understanding modern NLP systems and their role in advanced computational research.

1.2 Fundamental elements of NLP

NLP can be broadly divided into two main components: *Natural Language Understanding (NLU)* and *Natural Language Generation (NLG)*, which focus on comprehending and generating text, respectively. Figure 1.1 illustrates this high-level classification of NLP. The aim of this section is to provide an overview of NLU and NLG.

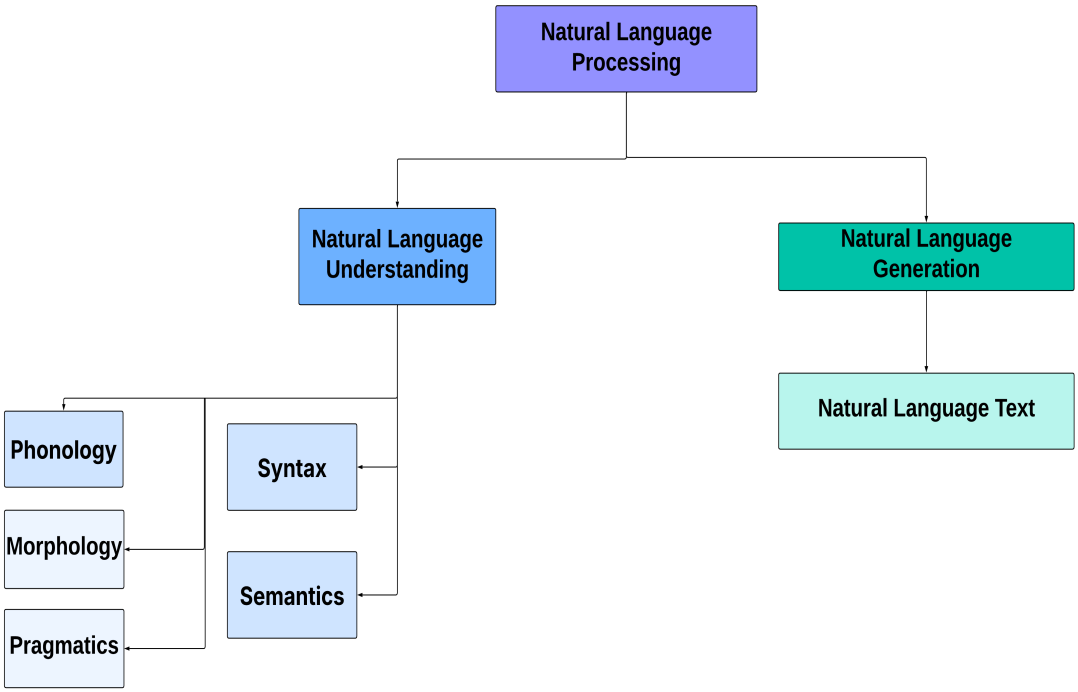


Figure 1.1: Broad classification of NLP.

1.2.1 NLU

NLU enables machines to comprehend human language by analyzing text and extracting key information such as concepts, entities, emotions, and keywords. It finds practical applications in customer support systems, helping to interpret issues reported by users either verbally or in written form. Linguistics, the scientific study of language, encompasses meaning, context, and the diverse structures of language. Therefore, understanding essential NLP terminologies and the various levels of NLP is crucial. In the following, we explain some common terms used at multiple levels of NLP.

a) Phonology

Phonology is a subfield of linguistics concerned with the systematic organization of sounds in language. The term derives from Ancient Greek, where *phono* means "voice" or "sound," and the suffix *-logy* denotes "study". According to Nikolai Trubetzkoy (1993), phonology is “the study of sound pertaining to the system of language”, while Lass (1998) [95] describes it more broadly as the study of the sounds of language, concentrating on their behavior and organizational patterns within linguistic systems.

b) Morphology

The components of a word is the minimal unit of meaning, known as *morphemes*. Morphology, which studies the structure and nature of words. For instance, the word *Preemption* can be decomposed into three morphemes: the root *empt*, the prefix *pre* and the suffix *-tion*. Since the meanings of morphemes tend to remain stable across different words, humans can often infer the meaning of unfamiliar terms by analyzing their constituent morphemes. Words that cannot be further divided and carry meaning independently are called *lexical morphemes* (e.g., *table*, *chair*). Morphemes that attach to lexical morphemes to modify meaning or function are called *grammatical morphemes* (e.g., *-ing*, *-ed*, *-ly*, *-est*, *-ful*), as seen in words like *consulting*, *worked*, *likely*, and *use*. Grammatical morphemes that occur only in combination with other morphemes are referred to as *bound morphemes* (e.g., *-ed*, *-ing*), which are divided into *inflectional* and *derivational* morphemes. Derivational morphemes alter a word's meaning or grammatical class; for instance, *real* becomes the verb *realize* with the addition of *-ize*. Inflectional morphemes modify a word's grammatical properties, such as gender, mood or aspect. For example, adding the inflectional morpheme *-ed* transforms the root *camp* into *camped*.

c) Lexical

At the lexical level, both humans and NLP systems focus on interpreting the semantic content of individual words. Various processing techniques contribute to word-level understanding, the primary one being the assignment of a part-of-speech (PoS) tag to each word. At the lexical level, semantic representations are often simplified to single-meaning words, although the exact representation depends on the semantic theory adopted in the NLP system. Lexical analysis examines word structure in relation to meaning and part of speech, while also segmenting text into sentences and words. Correct PoS tagging improves the comprehension of a sentence's intended meaning and supports tasks such as cleaning and feature extraction using techniques like stop word removal, stemming, and lemmatization. Stop words, such as '*in*', and '*the*', are eliminated because they contribute little semantic value and appear frequently, which can increase computational overhead. Stemming simplifies words by stripping away suffixes to obtain their root form; for example, *accounting* and *accountant* become *account* and *singer* reduces to *sing*. Lemmatization, in contrast, uses a vocabulary and linguistic rules to return the

proper base form of a word. For instance, the token *drived* would be reduced to *drive* by lemmatization, whereas stemming would produce *driv*. This distinction ensures that words are analyzed in their correct linguistic context.

d) Syntactic

After the lexical level, words are organized into phrases, which are then combined into clauses and ultimately assembled into complete sentences at the syntactic level. This stage focuses on ensuring the correct grammatical structure of sentences by analyzing the relationships between their components. The resulting output reveals the structural dependencies among words. This process, known as *parsing*, identifies meaningful phrases that convey richer information than individual words.

The syntactic level looks at how words are arranged and function grammatically, beyond what lexical analysis covers. Changing the order of words can modify the relationships between them and thereby influence the overall meaning of a sentence. For instance, the sentences "Sarah defeats Tom in a race " and "Tom defeats Sarah in a race" differ only in syntax, yet their meanings are entirely distinct [186]. Unlike lexical processing, stop words are retained at this level because removing them can alter the sentence meaning. Syntactic analysis does not employ stemming or lemmatization, as reducing words to their base forms can disrupt grammatical correctness.

e) Semantic

At this level, the main objective is to determine the precise meaning of a sentence. While humans rely on prior knowledge of language and concepts to interpret meaning, machines lack this innate ability. Semantic processing infers sentence meaning by analyzing its structure and the key words that express conceptual relationships. For example, a sentence may pertain to "movies" even if it does not explicitly include that word, but contains related terms such as "Screenplay", "Picture", "Cinema", or "film".

At this level, semantic ambiguities arising from words with multiple meanings are also resolved [99]. For instance, the word "bark" can refer either to the sound made by a dog or the outer covering of a tree, depending on the context. Semantic analysis considers both dictionary definitions and contextual usage to derive meaning. Correct interpretation is determined by examining the context provided by the rest of the sentence [58].

f) Discourse

The discourse level of NLP examines multiple sentences, analyzing their logical structure and establishing connections that ensure coherence. This level focuses on meaning-bearing properties of text by interpreting inter-sentential relationships and identifying linguistic structures across multiple text levels [99]. Two of the most commonly addressed tasks at this level are *Anaphora Resolution* and *Coreference Resolution*. The first one involves identifying the entity referred to by an anaphor, thereby resolving references within a text to maintain a consistent meaning. Coreference resolution, on the other hand, identifies all expressions in a text that refer to the same entity. This process is crucial for advanced NLP tasks, including document summarization and information extraction.

g) Pragmatic

This branch of NLP focuses on knowledge and information beyond the explicit content of a text. It examines sentences with implied or indirect meanings. Real-world knowledge is applied to interpret the intended meaning of a text. When a sentence is vague and the context fails to offer enough clarification, pragmatic ambiguity arises [190]. Pragmatic ambiguity occurs when different individuals interpret the same text differently, depending on the context and the reader's or listener's background knowledge.

Pragmatic analysis focuses on the *inferred meaning* derived from context. However, pragmatic analysis studies context and background knowledge to uncover the intended meaning of linguistic expressions, whereas semantic analysis studies the relationship between words and their literal meanings. The goal of NLP is to integrate various aspects of language understanding and generation within an algorithmic system. Such systems can even handle multilingual tasks. For instance, [149] proposed a modular cross-lingual event extraction system for English and Dutch, and Italian texts. Their framework enables the extraction of events, participants, locations, and times, along with their interrelations, generating event-centric knowledge graphs. Each module processes standardized input and produces output for the next module, and the modular design allows for flexible configuration, adaptation, and dynamic distribution. Ambiguity is a central challenge in natural language processing, occurring when a sentence can be interpreted in multiple ways. For example, semantic ambiguity arises when the meaning of words is unclear or

context-dependent, while lexical ambiguity occurs when a single word has multiple meanings. Resolving ambiguity often requires knowledge of the complete sentence. Several approaches have been proposed, including preserving ambiguity, minimizing ambiguity, weighting ambiguity and interactive disambiguation [164]. Methods such as preserving ambiguity [56, 63, 164] aim to maintain multiple interpretations where appropriate while employing statistical techniques to resolve uncertainties and improve understanding.

1.2.2 NLG

NLG refers to the task of generating a well-structured and meaningful text from an internal representation. As a key component of NLP, NLG typically occurs in 4 stages: identifying the communicative goals, developing a strategy to achieve these objectives by assessing the situation and available resources, and finally realizing the plan as natural language text . The Figure below 1.2 shows that NLG serves as the counterpart to NLU.

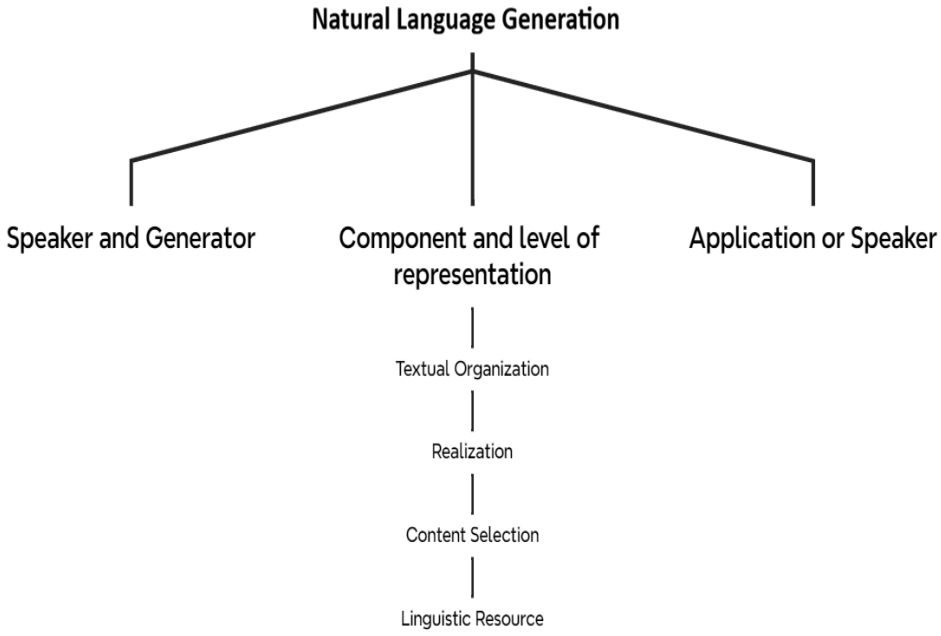


Figure 1.2: Components of NLG.

a) Speaker and Generator

Text generation requires both a speaker or application that holds the intended message, and a generator or program that expresses these intentions in a coherent language suited to their context.

b) Components and Levels of Representation

The language generation process involves several interrelated tasks.

- *Content Selection*: Relevant information must be identified and included in the output.
- *Textual Organization*: The selected information needs to be arranged according to grammatical rules, maintaining both sequential order and appropriate linguistic relationships such as modifications.
- *Linguistic Resources*: Suitable linguistic resources, including word choices, idioms, and syntactic structures, must be selected to support the realization of the information.
- *Realization*: Finally, the organized and resource-supported information is converted into an actual text.

c) Application or Speaker

During this phase, the speaker only begins the process and does not yet actively engage in generating language. The system stores the history, organizes potentially relevant content, and constructs a representation of its knowledge. This overall representation constitutes the situation, from which a subset of propositions is selected. The only condition is that the speaker has a coherent understanding of the situation [124].

1.3 NLP through time

Toward the end of the 1940s, the term *Natural Language Processing* had not yet been coined, but research on machine translation (MT) had already started. During this period, studies were not confined to a single region, with English and Russian being the primary languages used in MT research [7]. However, MT research nearly came to a

halt in 1966 following the ALPAC report, which concluded that MT was not progressing. Despite this, some MT production systems eventually deliver usable output to clients [82]. Concurrently, early efforts emerged to use computers in literary and linguistic analysis. By 1960, AI-influenced work emerged with systems such as the BASEBALL question-answering system [68]. Natural successors to these early systems included LUNAR [196] and Winograd's SHRDLU, which demonstrated more advanced linguistic and task-processing capabilities.

Progress in NLP was largely driven by two fronts: the ARPA Speech Understanding Research (SUR) project, along with major system development efforts focused on building database front ends [76]. These front-end initiatives seek to go beyond LUNAR by enabling more advanced interaction with large-scale databases. By the early 1980s, computational grammar theory had emerged as an active research area, incorporating logic-based representations of meaning, user beliefs and intentions, as well as linguistic functions such as emphasis and thematic roles.

By the late 1980s, robust general-purpose sentence processors such as SRI's Core Language Engine [5] and Discourse Representation Theory [87] facilitated the analysis of extended discourse using a grammatico-logical framework. During this period, research communities expanded, accompanied by the development of practical resources, grammars, and tools such as the Alvey Natural Language Tools becoming available [28]. The DARPA conferences on speech recognition and message understanding emphasized rigorous evaluation, establishing a trend that defined much of NLP research in the 1990s.

Research in user modeling [189], compositional theories of tune interpretation [41], and rhetorical schema-based text generation [115] marked significant advancements. Early work on word sense disambiguation [170], probabilistic networks, and lexicon studies laid the foundation for statistical approaches. By the 1990s, statistical language processing [109], information extraction, and automatic summarization [107] became major research focus.

1.3.1 An overview of recent advancements in NLP

NLP's main objectives are to understand, analyze, and process natural language data to achieve specific objectives using a variety of algorithms, tools, and techniques. However, numerous challenges arise due to the complexity and variability of natural language, making it challenging to meet all objectives through one approach. Consequently, significant research attention has been devoted to developing diverse tools and methods within NLP and related fields. Figure 1.3 provides an overview of the key developments described.

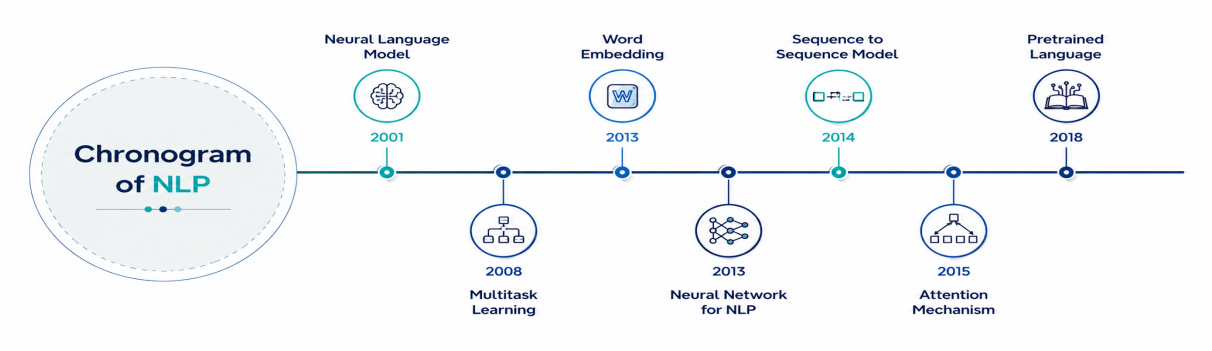


Figure 1.3: An overview of recent advances in NLP.

In the early 2000s, *neural language modeling* emerged, where the probability of the next word (also known as token) is predicted based on the preceding n words. [19] presented the idea of a feedforward neural network combined with a lookup table to represent these preceding words in sequence. *Multitask learning* in NLP applied by [42], using two convolutional models with max pooling for POS and named entity recognition (NER) tagging. [118] introduced *word embeddings*, introducing dense vector representations of text and highlighting limitations of traditional sparse bag-of-words models.

Following these advancements, neural networks gained traction in NLP, handling variable-length input sequences. Sutskever et al. [175] introduced a general sequence-to-sequence framework that employs encoder-decoder networks to transform sequences into vectors and subsequently reconstruct them back into sequences. Neural networks gradually transformed NLP, enabling significant developments in the field. Initially, CNNs were applied to image classification and visual data analysis, but later adapted for NLP tasks such as sentence classification [171], sentiment analysis [181], text classification [154], summarization [201], machine translation [104], and answer relation tasks [195]. Works

by Newatia (2019) [125], and Wang et al. [192] illustrate CNN architectures and their NLP applications.

Recurrent Neural Networks (RNNs), which process sequential data, have also been widely used in NLP, suitable for text, time series, speech, audio, and video [183]. A modified form, *Long Short-Term Memory (LSTM)* networks, selectively retain important information for long durations while discarding irrelevant data [69, 78]. The *Gated Recurrent Unit (GRU)*, a simpler variant of LSTM, has shown superior performance in many tasks [38, 39]. Additionally, *attention mechanisms* [11] and *transformers* [187] have significantly advanced NLP, with models like *BERT* [48] enabling powerful bidirectional contextual understanding. Recent surveys such as Otter et al. [130] summarize deep learning applications in NLP.

Many tools and systems have been developed to shape the modern NLP. These include *Sentiment Analyzers*, POS taggers, *chunking*, NER, *emotion detection*, and *Semantic Role Labeling (SRL)*.

- **Sentiment analysis** [199] extracts sentiments about a topic, involving feature extraction, sentiment identification, and relationship analysis. It relies on sentiment lexicons and pattern databases, rating documents on a scale from -5 to $+5$. While most POS tagsets are based on English and Indo-European languages, research has extended to languages like Arabic [204], Sanskrit [182], and Hindi [144]. Techniques such as treebank creation, suffix stripping, and SVM-based supervised learning have been used for accurate tagging [49].

- **Chunking** separates phrases from unstructured text, grouping meaningful multi-word expressions (e.g., "North Africa") as single units. Also called "shadow parsing," it labels syntactic structures such as verb phrases (VP) and noun phrases (NP). Chunking evaluation frequently uses CoNLL 2000 shared task datasets [114, 160, 174].

- **Named Entity Recognition (NER)** identifies words representing real-world entities (e.g., persons, locations, organizations). NER is particularly challenging in social media texts, which contain slang and non-standard language. Ritter [148]

adapted NLP pipelines for tweets, improving performance compared to standard NLP tools.

- **Emotion detection** analyzes emotions expressed in text, speech, or gestures. Sharma [162] studied Hinglish (mixed English-Hindi) conversations, detecting emotions using machine learning combined with human knowledge. Seal et al. [159] proposed a method leveraging an emotional keyword database, phrasal verbs, and negation analysis, achieving improved performance.

- **Semantic Role Labeling (SRL)** assigns semantic roles to words in a sentence, identifying arguments of verbs according to verb frames. State-of-the-art SRL systems involve parsing, argument identification, and classification to compute corresponding SRL tags [136].

- **Event detection in social media** uses graphical models to extract event information such as names, locations, and time from noisy data streams [21], aggregating multiple messages to create accurate event records.

1.4 Applications of NLP

NLP has applications in many fields. In the following section, we examine several of these areas and highlight relevant work in each.

1. Machine Translation. With the majority of the world now online, ensuring data accessibility to all users presents a significant challenge. One of the primary obstacles is the language barrier, as numerous languages exist with diverse sentence structures and grammatical rules. Machine Translation (MT) aims to convert text from one language to another, often using statistical engines such as Google Translate. The main difficulty in MT is preserving sentence meaning and proper grammar and tense. In 2016, Google introduced a neural network-based MT system utilizing deep learning techniques, representing a major advancement in the field.

To assess the quality of machine translation, several automatic evaluation metrics have been developed, which compare hypothesis translations with reference translations.

Common methods include Word Error Rate (WER), Position-Independent Word Error Rate (PER) [184], Generation String Accuracy (GSA) [12], Multi-Reference Word Error Rate [129], BLEU score [138], and NIST score [51]. These metrics aim to approximate human judgment and often show a high correlation with subjective human evaluations of fluency and adequacy.

2. Text Categorization. Categorization systems are designed to process large volumes of data, such as official documents, market data, and newswires, and automatically categorize them according to predefined classes or indices. For instance, the Carnegie Group's Construe system [75] processes Reuters articles, significantly reducing the workload of human staff or indexers.

In this task, two primary models are employed. [111], both assuming a fixed vocabulary. In the first, the Multi-variate Bernoulli model, a document is formed through the selection of a subset of vocabulary terms, with each selected word used at least once, regardless of order. This model captures which words are present in a document without considering word frequency or order. The second, the Multi-nomial model, produces a document by selecting words and arranging them in any order, thereby also capturing the frequency of each word.

3. Spam Filtering. Recently, a range of machine learning techniques have been employed for this task. These include Rule Learning [40], Naïve Bayes [8, 147, 151], Memory-Based Learning [153], Support Vector Machines [53], Decision Trees [29], Maximum Entropy Models [22], as well as Hash Forests and rule encoding methods [198], sometimes combining multiple learners [152]. Using these machine learning approaches is advantageous, as the classifiers are learned from training data rather than manually constructed. Naïve Bayes is particularly favored due to its simplicity and strong performance [96]. Traditionally, the majority of anti-spam email filtering systems have utilized the Multivariate Bernoulli model [24].

4. Information Extraction. (IE) focuses on identifying relevant phrases and entities within textual data such as locations, names, prices and dates. Hidden Markov Models (HMMs) are often employed to obtain information from research papers, enabling field-specific searches, effective presentation of results, and reference matching. A practical example

of information extraction can be seen in pop-up advertisements on e-commerce websites that display recently viewed items with discounts.

In this task, two models are commonly used [111] like the task of Text Categorization, the Multi-variate Bernoulli model and the Multi-nomial model.

Knowledge discovery has become an increasingly important research area. Techniques for extracting useful information from source documents include POS tagging, Chunking or Shadow Parsing, stopword removal, and stemming. Stemming maps words to their base forms and can be implemented via dictionary-based methods or algorithmic approaches [142], with a trade-off between accuracy and computational cost. Other techniques include indexing multi-token units through compound or statistical phrases, and Word Sense Disambiguation (WSD), which identifies the appropriate meaning of a word in context and can replace terms with their corresponding senses in document vectors.

Extracted information can be applied in various ways, such as generating summaries, building databases, identifying keywords, or classifying text into predefined categories. For example, the CONSTRUE system, developed for Reuters, classifies news stories [75]. Although many IE systems can effectively extract terms, identifying the relationships between them continues to be a challenge. PROMETHEE is a system designed to extract lexico-syntactic patterns corresponding to specific conceptual relations [123]. Effective IE systems operate at multiple levels, from word recognition to discourse analysis of entire documents. The Blank Slate Language Processor (BSLP) [26] demonstrates this approach by analyzing real-world natural language corpora, such as responses to open-ended advertising questionnaires.

An example of information extraction from life insurance applications is MetLife's Intelligent Text Analyzer (MITA) [64]. Ahonen et al. [2] further introduced a comprehensive text-mining framework integrating pragmatic and discourse-level analysis to improve textual comprehension.

5. Summarization. This task consists of summarizing large volumes of data while preserving their meaning increasingly important. Text summarization not only allows us to identify and understand key information from extensive datasets but also helps in capturing deeper emotional insights. For instance, companies can gauge general sentiment

on social media to inform strategies for new product launches, making sentiment analysis a valuable marketing tool.

Text summarization can be classified according to the number of documents considered, primarily into single-document summarization and multi-document summarization. Additionally, summaries can be classified as either generic or query-focused [66, 54, 191, 135]. Summarization tasks may follow supervised or unsupervised approaches. Supervised systems require training data to select relevant content from documents, which often involves large amounts of annotated data for effective learning.

6. Dialogue System. Dialogue systems have grown increasingly significant in practical applications, encompassing everything from offering user support to performing specific tasks. Support-oriented dialogue systems require context awareness to respond appropriately, whereas task-oriented systems often operate effectively with minimal contextual information. Early dialogue systems were typically designed for small-scale applications, like controlling home theater systems. Modern, more sophisticated dialogue systems, often referred to as "habitable" systems, leverage all levels of language processing to enable fully automated interactions. These advancements have paved the way for robots and virtual assistants to communicate with humans naturally. Prominent examples include Google Assistant, Apple's Siri, Windows Cortana and Amazon Alexa, which demonstrate the potential of dialogue systems in everyday applications.

7. Medicine. NLP has also been extensively applied in the medical domain. One of the prominent large-scale projects is the Linguistic String Project-Medical Language Processor (LSP-MLP) [32, 70, 77, 105]. By extracting and summarizing information about symptoms, signs, patient responses and drug dosages , the LSP-MLP helps physicians identify potential side effects of medications while highlighting relevant data items [150]. The Centre d'Informatique Hospitalière de Genève has worked on an electronic archiving system with NLP capabilities [113, 155], initially archiving patient records. Later, the LSP-MLP was adapted for French [14, 105, 126], culminating in the development of the RECIT system [13, 14], which employed a method called Proximity Processing [122]. RECIT was designed as a robust, multilingual NLP system capable of analyzing medical sentences[146, 145].

1.5 Word Embeddings

Word embeddings are numerical vector representations of words designed to capture semantic and syntactic relationships between terms. Unlike traditional text representation techniques such as Bag-of-Words (BoW) or TF-IDF, which represent words as discrete symbols, word embeddings map words into a continuous low-dimensional vector space where semantically similar words are located close to each other.

The main objective of word embeddings is to enable machine learning and deep learning models to understand linguistic relationships between words. For example, embeddings can capture similarities such as: $kingman + womanqueen$ This property demonstrates that embedding models are capable of learning semantic structures from large text corpora.

1.5.1 Types of Word Embedding Methods

Word embedding methods can generally be classified into three main categories: count-based methods, predictive neural methods, and contextual embeddings.

Count-Based Methods

Count-based approaches rely on statistical information extracted from word occurrences in a corpus.

1. Bag-of-Words (BoW) represents text using word occurrence frequencies. Each document is transformed into a sparse vector where each dimension corresponds to a vocabulary term. Although BoW is simple and computationally efficient, it suffers from several limitations such as: high dimensionality, sparse representations, lack of semantic understanding, ignoring word order and context.

2. TF-IDF (Term Frequency–Inverse Document Frequency) improves BoW by assigning weights to words according to their importance in a document relative to the entire corpus. The TF-IDF score is computed as:

$$TF\text{-}IDF(t, d) = TF(t, d) \times IDF(t)$$

where: $TF(t,d)$ is the term frequency and $IDF(t)$ is the inverse document frequency. This method reduces the influence of common words but still fails to capture semantic relationships between words.

Predictive Neural Embedding Methods

Predictive methods use neural networks to learn dense vector representations of words.

1. Word2Vec, proposed by Mikolov et al. [118], is one of the most influential word embedding techniques. It learns vector representations by predicting words from their surrounding context. Word2Vec consists of two architectures: Continuous Bag-of-Words (CBOW) and Skip-Gram CBOW predicts a target word based on surrounding context words, whereas Skip-Gram predicts surrounding words from a target word. The Skip-Gram objective function is defined as:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

where: T is the total number of words, c is the context window size and w_t is the target word. Word2Vec generates dense vector representations capable of capturing semantic and syntactic similarities between words.

2. GloVe(Global Vectors for Word Representation), proposed by Pennington et al. [140], combines matrix factorization and local context learning using global word co-occurrence statistics. GloVe effectively captures semantic relationships between words but requires large co-occurrence matrices, increasing memory requirements.

3. FastText, introduced by Bojanowski et al. [25], extends Word2Vec by representing words as collections of characters n-grams. This approach enables FastText to: handle rare words, represent unseen words and perform effectively on morphologically rich languages. However, FastText generally requires more computational resources than Word2Vec.

Contextual Embeddings

Traditional embeddings assign a single vector representation to each word regardless of context. Contextual embedding overcomes this limitation by generating context-dependent representations.

1. ELMo (Embeddings from Language Models) uses bidirectional Long Short-Term Memory (BiLSTM) networks to generate contextualized embeddings [141].

2. BERT (Bidirectional Encoder Representations from Transformers) , introduced by Devlin et al. [48], employs Transformer architectures and bidirectional learning to produce highly contextualized word representations. Although BERT achieves state-of-the-art performance in many NLP tasks, it requires substantial computational resources and large training datasets.

1.5.2 Comparison of Word Embedding Methods and discussion

In this subsection, we present a comparative analysis of the word embedding methods discussed above.

Table 1.1: Comparison of Word Embedding Methods

Method	**Type**	**Contextual**	**Advantages**	**Limitations**
BoW	Count-based	No	Simple and fast	Sparse vectors, no semantic understanding
TF-IDF	Statistical	No	Highlights important words	No contextual understanding
Word2Vec	Neural	No	Efficient, captures semantic similarity, low dimensionality	One vector per word
GloVe	Neural-statistical	No	Captures global co-occurrence statistics	Large memory requirements
FastText	Neural	No	Handles rare and unseen words	Higher computational cost
ELMo	Deep Learning	Yes	Context-aware embeddings	Computationally expensive
BERT	Transformer	Yes	Excellent contextual understanding	Very high computational complexity

In this research, Word2Vec was selected as the word embedding technique for several reasons.

First, Word2Vec provides dense vector representations capable of capturing semantic and syntactic relationships between words efficiently. Unlike traditional methods such as BoW and TF-IDF, Word2Vec preserve semantic similarity by placing related words close to each other in the vector space.

Second, Word2Vec offers a good balance between embedding quality and computational efficiency. Compared with advanced contextual models such as BERT and ELMo, Word2Vec requires significantly lower computational resources and shorter training times, making it suitable for environments with limited hardware capabilities.

Additionally, Word2Vec performs effectively even with moderately sized datasets, whereas contextual models generally require massive corpora and expensive computational infrastructure.

Another important motivation is the simplicity and interpretability of Word2Vec. Its architecture is relatively straightforward, facilitating implementation, parameter tuning, and integration into downstream NLP tasks.

Furthermore, the Skip-Gram architecture is particularly effective in learning semantic relationships between words appearing in similar contexts, which improves textual feature representation in the proposed research.

For these reasons, Word2Vec was considered the most appropriate embedding method for this study, providing an effective compromise between semantic representation quality, computational efficiency, and implementation simplicity.

1.6 Conclusion

This chapter presented a comprehensive overview of NLP, including its fundamental concepts, historical evolution, and major application domains. It introduced the two core components of NLP: NLU, which focuses on interpreting and extracting meaning from human language, and NLG, which aims to produce coherent and contextually appropriate text. Together, these components form the foundation of modern intelligent language-processing systems.

The chapter also examined the historical development of NLP, from early rule-based and machine translation systems to probabilistic and statistical approaches. Important milestones such as BASEBALL, LUNAR, and SHRDLU, along with advances in computational linguistics and discourse representation, contributed significantly to the evolution of the field. The transition toward statistical and machine learning methods during the late 1980s and 1990s laid the groundwork for modern NLP applications.

Furthermore, the chapter discussed the emergence of deep learning techniques in NLP, including neural language models, word embeddings, CNNs, RNNs, LSTMs, GRUs, attention mechanisms, and transformer architectures such as BERT. These approaches significantly improved contextual language understanding and enabled advanced applications including sentiment analysis, named entity recognition, emotion detection, and semantic role labeling.

Several important NLP applications were also reviewed, including machine translation, text categorization, spam filtering, information extraction, summarization, dialogue systems, and medical applications, demonstrating the growing impact of NLP in both research and industry. In addition, the chapter analyzed different text embedding methods, ranging from traditional techniques such as Bag-of-Words and TF-IDF to neural and contextual embeddings such as Word2Vec, GloVe, FastText, ELMo, and BERT. Considering the balance between performance, simplicity, and computational efficiency, Word2Vec was selected as the most suitable embedding technique for this research.

Despite the significant progress achieved in NLP, several challenges remain, including high computational complexity, scalability issues, semantic ambiguity, and the difficulty of efficient model training. These limitations motivate the exploration of alternative computational paradigms capable of improving intelligent language-processing systems.

Therefore, the next chapter introduces quantum computing, a novel computational framework based on the principles of quantum mechanics. By exploiting phenomena such as superposition and entanglement, quantum computing offers promising capabilities for complex data processing and optimization tasks. Understanding these concepts is essential before exploring the integration of QML techniques with NLP applications, which represents the central objective of this thesis.

Chapter 2

Quantum Computing

2.1 Introduction

Quantum refers to the minimal discrete unit of a physical quantity, such as energy or mass. In 1900, Max Planck proposed that energy at atomic and subatomic scales is not continuous but instead composed of discrete packets called quanta [57]. A key concept underlying this idea is wave–particle duality, which describes how matter and light can exhibit wave-like behavior in some contexts and particle-like behavior in others.

This chapter introduces the fundamental concepts of quantum computing and provides the theoretical background necessary for understanding quantum-based computational models. It first presents the core principles of quantum information processing, including qubits, superposition, the observer effect, entanglement, quantum gates, measurement, and quantum circuits. These concepts form the basis of quantum computation and explain how quantum systems manipulate and encode information.

The chapter then explores several important quantum algorithms that have demonstrated the potential capabilities of quantum computing. These include Shor's algorithm, Grover's algorithm, the Quantum Fourier Transform (QFT), and the Variational Quantum Eigensolver (VQE). In addition, the chapter discusses Quantum Machine Learning (QML) algorithms, which combine quantum computing techniques with machine learning methods to address complex computational and data-processing tasks.

2.2 Basic concepts

2.2.1 Qubit

The QuBit is a two-dimensional complex vector that corresponds to a point on the surface of the Bloch sphere [89], as illustrated in Figure 2.1. It can be interpreted as a "fuzzy" combination of the classical states 0 and 1, reflecting its ability to exist in superposition. In principle, a qubit may be realized by any physical system that exhibits quantum behavior, such as an electron or an atom. Thus, a qubit can be expressed as:

- An electron in an atomic orbital, with 0 representing the ground state and 1 the excited state.
- A photon, represented as a two-level system in which 0 and 1 correspond to its polarization states.

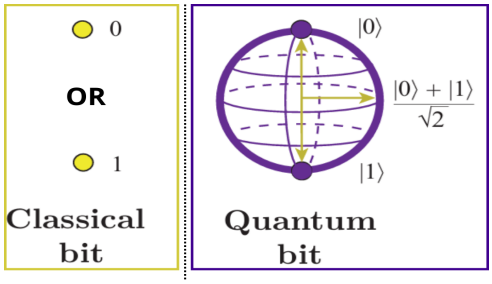


Figure 2.1: Classical bit and quantum bit.

In reality, qubits used in today's computing industry are typically realized as tiny superconducting circuits fabricated using nanotechnology, capable of oscillating in two directions. Their quantum nature allows them to exist in superposition, making them inherently suitable for parallelism. This is among the principal reasons why quantum computers have the potential to achieve exponential gains in computational power compared to classical, bit-based systems. Qubits are the basic units of quantum information and represent the quantum counterpart of classical binary bits, but with the added capability of representing both logical states simultaneously through superposition. A qubit is a two-level quantum mechanical system. The fundamental differences between a classical

bit and a qubit are illustrated in Figure 2.2. From a mathematical perspective, a qubit is described by a normalized two-dimensional state vector describing a quantum system with two basis states. Two coefficients, α and β describe this representation, where α is real and β is complex. A qubit can be represented as shown in Equation 2.1 by a vector which is expressed in Dirac notation and is referred to as a ket.

$$|\psi\rangle = \begin{bmatrix} c_0 & c_1 \end{bmatrix}^T, \text{ where: } |c_0|^2 + |c_1|^2 = 1. \tag{2.1}$$

The ket encodes the quantum state of the qubit, specifying the probability amplitudes associated with finding the qubit in each of its basis states [110]. Multiple qubits can

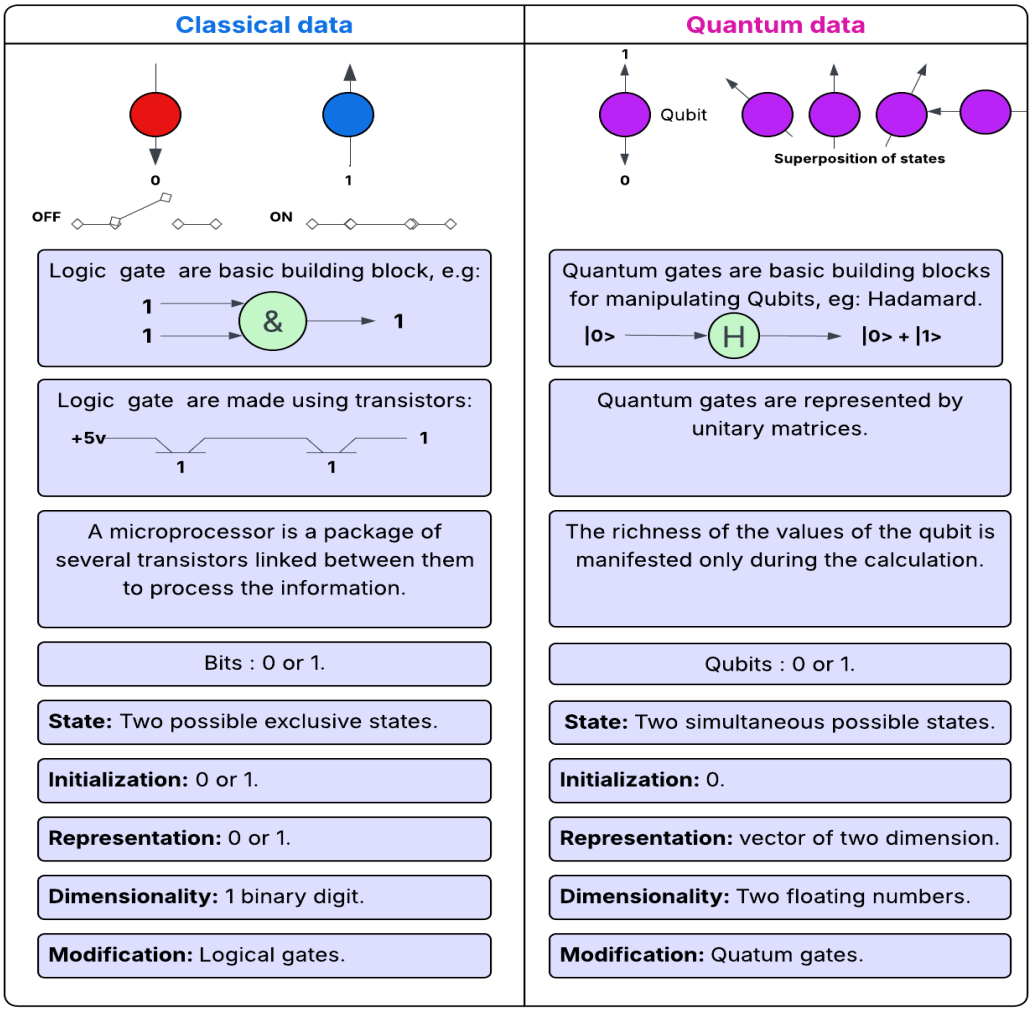


Figure 2.2: Classical bit Vs Qubit.

be combined to create a larger quantum system. The system's possible states arise from

the tensor product of its qubit states, capturing all combinations of their basis states. Accordingly, the state of the composite system can be expressed by the following ket:

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes ... \otimes |\psi_N\rangle \,. \tag{2.2}$$

2.2.2 Superposition

According to this principle of superposition [161], if a particle (or system) can exist in several possible states represented by the vectors $|\psi_n\rangle$, then, it may occupy any complex linear superposition of these states:

$$|\psi\rangle = \sum_{i}^{N} \alpha_i . \, |\psi_i\rangle \,. \tag{2.3}$$

However, superposition on its own is not what makes quantum computing uniquely powerful. A classical analog computer could, in principle, allow its bits to take continuous values between 0 and 1, but such a device would not be significantly more efficient than a standard digital computer. The true power of quantum computing arises from a particular kind of superposition that enables a quantum system to represent an exponentially large number of logical states simultaneously [83].

This implies that a system of qubits can simultaneously occupy multiple states as expressed in Equation 2.3. This property constitutes the first fundamental distinction from classical computing. It is precisely this principle that allows quantum bits to represent not only the classical values 0 and 1, but also any linear combination of these states—configurations known as superposition states [4].

2.2.3 Observer effect

To grasp the concept of the observer effect, it is necessary to examine quantum states and their associated probabilities more closely. The observer effect states that, prior to measurement, a qubit exists in a superposition of 0 and 1. However, once a measurement is performed, the qubit collapses to either state 0 or state 1. Outcome probabilities depend entirely on the qubit's quantum state and can follow any valid probability distribution [161]. As shown earlier in Equation 2.1, quantum state vectors can be expressed using Dirac's ket notation. For example, the state 0 and 1 are represented as $|0\rangle = [1, 0]^T$, and

$|1\rangle = [0, 1]^T$ respectively. By considering that the normalization condition in Equation 2.1 is satisfied, these two quantum states can be combined linearly. This condition ensures that the amplitudes of the state vectors correspond to valid probabilities, meaning that the squared magnitudes must sum to 100% probability, which is ensured by weighting the states in:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{bmatrix} 1.\alpha + 0.\beta \\ 0.\alpha + 1.\beta \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}. \tag{2.4}$$

These coefficients must satisfy that $\alpha^2 + \beta^2 = 1$. For example, a qubit in such a superposition possesses a probability of $\alpha^2 = 50\%$ of being measured in either the state $|0\rangle$ or the state $|1\rangle$.

$$|\psi\rangle = \frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle = \begin{bmatrix} 1.\frac{1}{\sqrt{2}} + 0.\frac{1}{\sqrt{2}} \\ 0.\frac{1}{\sqrt{2}} + 1.\frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}. \tag{2.5}$$

If a qubit is required to be in a superposition state corresponding to a 25% probability of measuring $|0\rangle$ and a 75% probability of measuring $|1\rangle$, the normalization condition $\alpha^2 + \beta^2 = 1$ must be satisfied. Setting $\alpha^2 = \frac{1}{4}$ yields $\alpha = \frac{1}{2}$, while $\beta = \frac{\sqrt{3}}{2}$. The resulting quantum state is shown in Equation 2.6.

$$|\psi\rangle = \frac{1}{2}\,|0\rangle + \frac{\sqrt{3}}{2}\,|1\rangle = \begin{bmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix}. \tag{2.6}$$

2.2.4 Quantum interference

To understand the role of quantum interference, it is helpful to first consider its classical analogue: noise cancellation.

Noise cancellation works by applying the principles of superposition and interference to suppress unwanted signals. This is achieved by producing a secondary signal matching the noise in frequency and amplitude, with a phase shift of $(2k + 1)\pi$, as illustrated in Figure 2.3. When these signals are superposed, destructive interference occurs, producing an output in which the noise is significantly reduced relative to the original signal. For a more detailed discussion, see [4, 106].

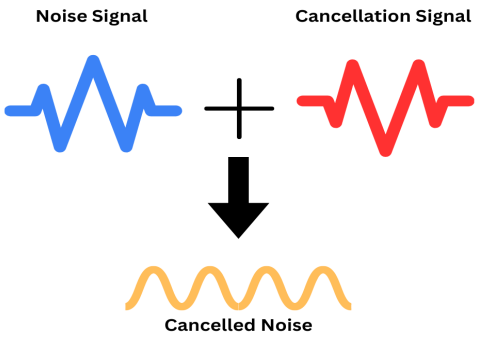


Figure 2.3: Example of classical noise cancellation.

Although noise cancellation in classical systems is implemented using digital circuits, the amplitude and phase of real-world signals are continuous variables, and perfect matching is impossible. As a result, the noise can never be fully eliminated. In quantum computing, a similar idea is applied, but instead of manipulating classical signals like sound waves, the system operates on quantum states. In this setting, interference arises from preparing a superposition of all possible computational states, see Figure 2.4.

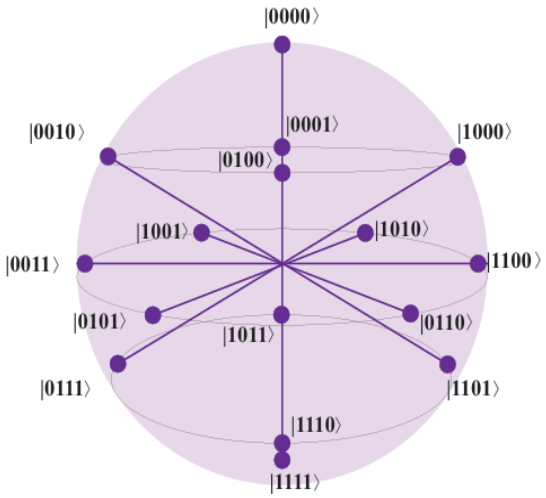


Figure 2.4: Superposition of multiple quantum states.

This superposition is then fed into a quantum circuit. Through this process, unwanted components are suppressed, while the amplitudes corresponding to the desired solution are enhanced. This selective interference enables the quantum circuit to produce the correct output [4].

In summary, quantum interference enables the steering of a quantum system toward desired outcomes by using destructive interference to suppress incorrect states and constructive interference to amplify those corresponding to the correct solution [106].

2.2.5 Entanglement

It occurs when quantum particles are correlated so that their states are defined only in relation to each other, even when they are widely separated. When a measurement is performed on one particle of an entangled pair and it is observed, for example, to be in the spin-down state, the other particle is instantaneously projected into the corresponding opposite spin-up state.

Quantum entanglement enables qubits, even those that are spatially distant, to exhibit instantaneous correlations. Consider a system consisting of two entangled electrons, denoted by A and B, which are separated by a large spatial distance. If an interaction or measurement affects only one electron, say particle A, the quantum state of the entire system is altered. Consequently, the probability distribution of the spin states of particle B changes immediately upon measuring the spin of particle A.

Thus, an external influence on one part of an entangled system impacts the system in its entirety. In general, the measurement or disturbance of one entangled particle affects the correlated properties of its partner. The entanglement is of particular importance in quantum information processing, as it enables the exploitation of wave particle duality and allows qubits to interact through interference effects in quantum algorithms.

2.2.6 Quantum gates

A central notion in quantum computation is that of quantum gates. These gates play a role similar to classical logic gates but operate on quantum information. They implement primitive, reversible transformations on qubits, preserving the information encoded in their quantum states [106]. Quantum gates form the fundamental building blocks of quantum circuits and, consequently, of all quantum computational processes. To understand their importance, it is useful to recall how computation works in the classical setting: classical computers operate by sending electrical pulses through circuits. A re-

ceived pulse at a specific time is interpreted as $True$ or 1, whereas the absence of a pulse is interpreted as $False$ or 0. All computations in this work are based on quantum circuit operations, which allow for more complex and sophisticated processing compared to classical computation.

A classical bit can exist in one of two states: 0 or 1, representing False and True, respectively, Whereas a classical bit has a definite value, a qubit can occupy a coherent superposition of $|0\rangle$ and $|1\rangle$, parameterized by complex amplitudes. In this superposition, the qubit is neither strictly 0 nor 1 until a measurement is conducted. Upon measurement, the qubit collapses to either the $|0\rangle$ or $|1\rangle$ state, with the probabilities of each outcome given by the squared magnitudes of the corresponding amplitudes, $|\alpha|^2$ and $|\beta|^2$. Representing this information computationally requires encoding quantum states as two-component complex vectors, which are significantly more intricate than the simple Boolean or binary values 0 and 1. Consequently, while classical systems can readily apply basic logical operations such as *not* or *or* to Boolean inputs, such straightforward operations are not applicable in the quantum setting, where states correspond to probability distributions encoded within complex vectors.

Bit Flip Pauli Gate (X-Gate)

By acting on a single qubit, this gate transforms its state from $|0\rangle$ to $|1\rangle$ or from $|1\rangle$ to $|0\rangle$. The X-Gate is represented by a 2×2 unitary matrix, which is used to manipulate the vector state of a qubit. Its action on a quantum state, as shown in equation 2.7, is a unitary transformation, Thus, the state vector norm is maintained and the total probability conserved. This gate is a key element in quantum algorithm development and logical operation implementation.

$$X.|\psi\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} . \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} 0.c_0 + 1.c_1 \\ 1.c_0 + 0.c_1 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_0 \end{bmatrix} . \tag{2.7}$$

Hadamard Gate (H-Gate)

This gate is a fundamental quantum logic gate that enables the creation of superposition states. It converts a qubit from the ground states $|0\rangle$ or $|1\rangle$ into a superposition of both.

This gate is represented by a matrix which is derived from the outer product of the states $|+\rangle$ and $|-\rangle$ with the computational basis vectors, as expressed in equation 2.8.

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \qquad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}. \tag{2.8}$$

The states $|+\rangle$ and $|-\rangle$ are quantum superposition states. The $|+\rangle$ state corresponds to an equal superposition of the basis states $|0\rangle$ and $|1\rangle$, whereas the $|-\rangle$ state represents a superposition with opposite relative phase. These states are essential to quantum computation, enabling the implementation of quantum operations.

$$H = |+\rangle\langle 0| + |-\rangle\langle 1| = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \text{ where : } |a\rangle\langle b| = \begin{bmatrix} a_0 \\ a_1 \\ . \\ . \\ . \\ a_n \end{bmatrix} . \begin{bmatrix} b_0 & b_1 & . & . & . & b_n \end{bmatrix}. \tag{2.9}$$

The H-Gate is extensively utilized in quantum algorithms to generate superposition states and to facilitate quantum search and classification procedures, see equations 2.9 and 2.10.

$$H.|0\rangle = \frac{1}{\sqrt{2}}. \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} . \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}. \tag{2.10}$$

Rotation Gates

Rotation gates are used to control the probability amplitudes of quantum states, enabling precise control over their evolution. From equation 2.4, the amplitude of a qubit's probability distribution can be set during its initialization. To adjust the probabilities of measuring 0 or 1 after initialization, it is necessary to apply Rotation Gates. To control the probability distribution of a qubit at initialization, instead of directly specifying the probabilities, one can define them using the angle θ represents the separation between $|0\rangle$ and $|\psi\rangle$ on the Bloch sphere (Figure 2.5) and determines the qubit vector's orientation, corresponding to the probability amplitudes.

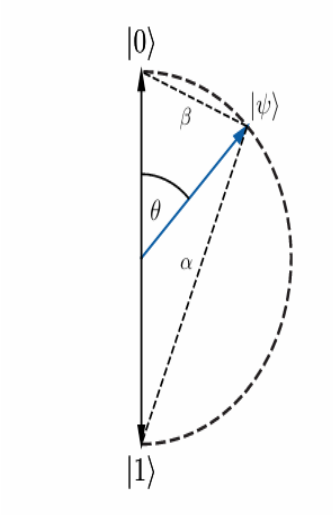


Figure 2.5: Schematic illustration of a quantum state using angular coordinates.

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}|1\rangle = \begin{bmatrix} \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2} \end{bmatrix}. \tag{2.11}$$

This interpretation can be used to rotate the qubit's state and thereby modify its probability distribution. Rotating $|0\rangle$ by θ transforms it into $|\psi\rangle$, and this part of the transformation is represented by the operator $|\psi\rangle\langle 0|$.

Similarly, the state of $|\psi'\rangle$ corresponds to the rotation of the state $|1\rangle$ by the angle θ. This process is shown in Figure 2.6.

The Equation 2.12 shows the matrix that represents a rotation operation.

$$R_y = |\psi\rangle\langle 0| + |\psi'\rangle\langle 1| = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}. \tag{2.12}$$

The R_y gate performs a rotation of a qubit around the y-axis. Its first parameter is the rotation angle θ in radians, where $\theta = 2\pi$ corresponds to a full 360° rotation. The second parameter specifies the qubit to which the gate is applied. It is important to note that the rotation does not halt upon reaching the $|1\rangle$ state; further rotation can actually reduce the probability of measuring $|1\rangle$. Similarly, the R_x gate rotates the qubit around the x-axis, as described by the matrix in Equation 2.13, while the R_z gate performs rotation around the z-axis, represented by the matrix in Equation 2.14.

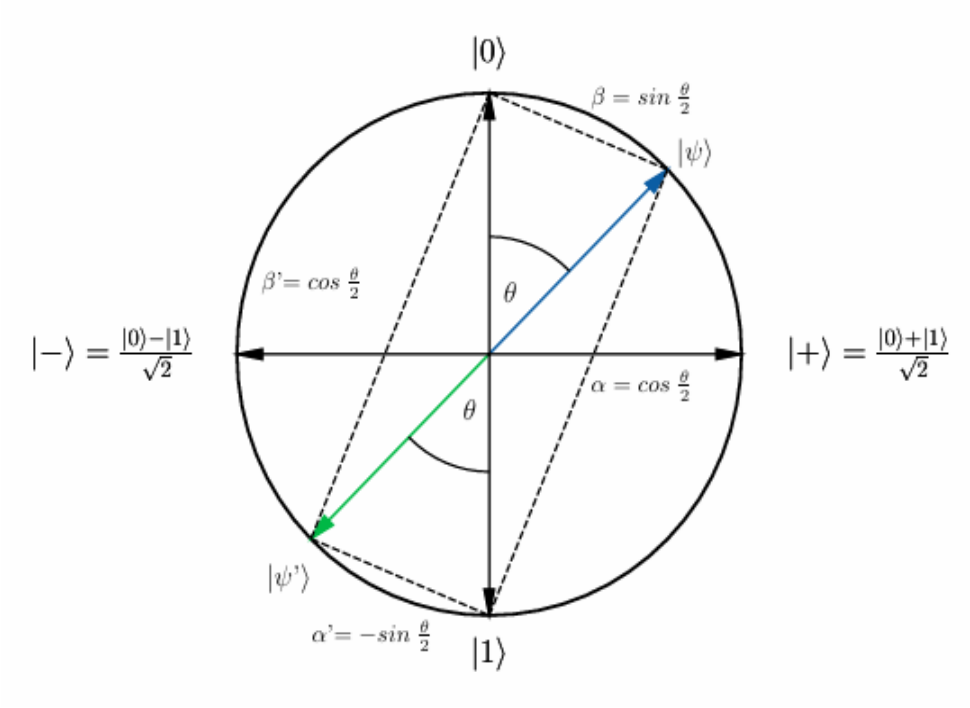


Figure 2.6: Illustration of states rotations on Bloch Sphere.

$$R_x = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}. \tag{2.13}$$

$$R_z = \begin{bmatrix} e^{-i\frac{\phi}{2}} & 0 \\ 0 & e^{i\frac{\phi}{2}} \end{bmatrix}. \tag{2.14}$$

Unitary Gate (U-Gate)

This gate is a type of unitary operator which is characterized by these parameters: a global phase and two angles that identify the qubit's position on the Bloch sphere. This gate is represented as shown in equation 2.15, for more details see [83].

$$U(\theta,\alpha,\beta) = \begin{bmatrix} \cos\frac{\theta}{2} & -e^{i\beta}\sin\frac{\theta}{2} \\ e^{i\theta}\sin\frac{\theta}{2} & e^{i\frac{\alpha+\beta}{2}}\cos\frac{\theta}{2} \end{bmatrix}. \tag{2.15}$$

C-NOT Gate

This gate corresponds to the tensor product of two operators A and B, as shown in equation 2.16, where A_{jk} and B_{lm} denote the matrix elements of A and B, respectively.

$$A \otimes B = \begin{pmatrix} A_{00}\begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} & A_{01}\begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} \\ A_{10}\begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} & A_{11}\begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} \end{pmatrix}. \tag{2.16}$$

$$\text{With } v = \begin{bmatrix} v_0 \\ v_1 \\ . \\ . \\ . \\ v_n \end{bmatrix} \text{ and } w = \begin{bmatrix} w_0 \\ w_1 \\ . \\ . \\ . \\ w_n \end{bmatrix} \tag{2.17}$$

$$\text{Then } v \otimes w = \begin{bmatrix} v_0w_0 & v_0w_1 & ... & v_0w_n & v_1w_0 & v_1w_1 & ... & v_1w_n & ... & v_nw_n \end{bmatrix}$$

Consider two qubits $|a\rangle$ and $|b\rangle$, represented by the following states $|a\rangle = a_0|0\rangle + a_1|1\rangle$ and $|b\rangle = b_0|0\rangle + b_1|1\rangle$. The combined state of this two-qubit system can be expressed using the tensor product of the individual states, as shown in equation 2.18, and represented in the form given in equation 2.17 by Mac-Kay Cisternas [106].

$$|ab\rangle = |a\rangle \otimes |b\rangle = a_0b_0|0\rangle|0\rangle + a_0b_1|0\rangle|1\rangle + a_1b_0|1\rangle|0\rangle + a_1b_1|1\rangle|1\rangle = \begin{bmatrix} a_0b_0 \\ a_0b_1 \\ a_1b_0 \\ a_1b_1 \end{bmatrix}. \tag{2.18}$$

This two-qubit gate can invert the state of the target qubit based on the state of the control qubit. The effect of the CNOT gate acting on qubits s_0 and s_1 is defined by the transformation shown in Equation 2.19. Here, X corresponds to the Pauli-X operator, while I denotes the identity operator that preserves the state of a qubit, for more details

see [4].

$$C_x s_0, s_1 = I \otimes |0\rangle\langle 0| + X \otimes |1\rangle\langle 1| = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}. \tag{2.19}$$

The CNOT gate does not alter the target qubit when the control qubit is $a|0\rangle$, and flips it when the control qubit is $a|1\rangle$. This gate is a reversible operator capable of implementing any unitary transformation within a quantum system. In addition, it constitutes a core component of many quantum algorithms, including Grover's algorithm and quantum factorization.

2.3 Measurements

This concept involves extracting information about the state of a quantum system. Qubits may exist in a superposition of states, and the act of measurement causes the superposition to collapse probabilistically into one of the basis states.

Similar to quantum gates, measurement can be regarded as an operation performed on qubits. Measurements are typically carried out on the standard basis, also known as the Z-basis or the classical computational basis. On this basis, the states $|0\rangle$ and $|1\rangle$ correspond to the classical logical values 0 and 1, respectively. Measurement operations can be combined with quantum gates to perform various types of observations. It is important to note that once a qubit is measured, its state becomes deterministic and cannot be altered unless a subsequent quantum operation is applied, see [83].

In contrast to quantum gates, measurement retrieves only partial information about a quantum state and typically eliminates phase information, enabling the conversion of quantum information into a classical bit. This process is the standard method of transferring information from a quantum system to a classical device, for more details, see [4].

2.4 Quantum circuits

Similar to classical computing, which relies on logical operations on bits, quantum computing is constructed from operations on qubits arranged within a circuit, referred to as a quantum circuit. This circuit [4] comprises a sequence of quantum operations applied to qubits, combined with measurement and preprocessing of classical information prior to encoding it into qubits. Below, Figure 2.7 shows a general architecture of the quantum circuit.

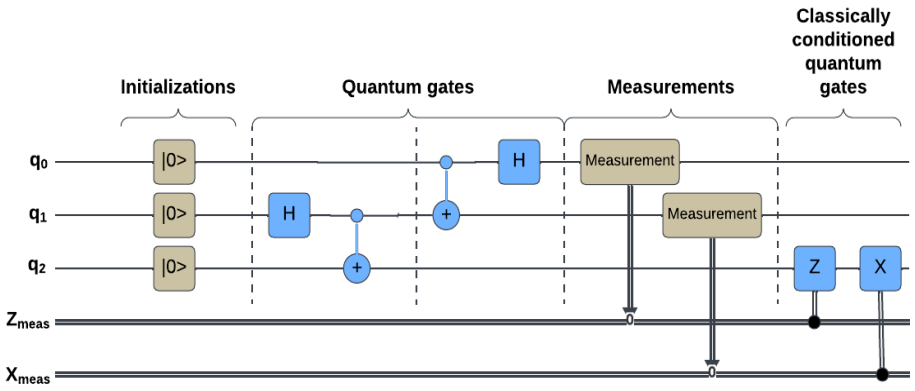


Figure 2.7: Quantum circuits.

Each horizontal line corresponds to a qubit. The left side of the circuit represents the quantum information initially encoded in the qubits, while the rightmost section denotes the quantum outputs resulting from the computations performed within the circuit. The operations applied to the qubits are illustrated as boxes, which may represent either quantum gates or measurement processes.

In summary, a quantum circuit allows classical information to be encoded in a quantum computer into quantum states, leverage fundamental quantum phenomena and ultimately convert the resulting quantum states into classical outcomes that provide solutions to the problem at hand.

2.5 Key quantum algorithms

2.5.1 Shor's algorithm

Shor's factoring algorithm [100] can be understood as a period-finding procedure in disguise. It is well-known for its ability to factor integers in polynomial time. Suppose we aim to factor a composite number N. First, we select an integer n such that $\gcd(n, N) = 1$, i.e., n must be coprime with N. The next step is to determine the period of the function $f(x) = n^x \bmod N$. In other words, we need to find an integer r such that $n^r \equiv 1 \pmod{N}$. It is important that r is even; if it is not, we must choose a different n. For an even r, we can proceed as follows:

$$(n^{\frac{r}{2}})^2 = 1 \mod N$$
$$(\mathrm{n}^{\frac{r}{2}} - 1)(n^{\frac{r}{2}} + 1) = 0 \mod N.$$

This implies that either $n^{\frac{r}{2}} - 1$, $n^{\frac{r}{2}} + 1$, or both share nontrivial factors with N. Consequently, the factors of N can be obtained as

$$\gcd(n^{\frac{r}{2}} - 1, N) \quad \text{or} \quad \gcd(n^{\frac{r}{2}} + 1, N),$$

or both. To implement this procedure, we require an oracle U_f that evaluates the function $f(x) = n^x \bmod N$. The corresponding quantum circuit is illustrated below:

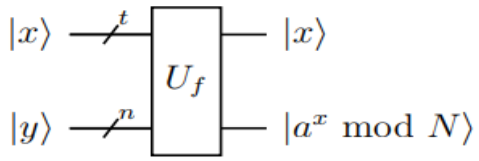


Figure 2.8: An oracle implementing the modular exponentiation function $f(x) = n^x \bmod N$.

One way to think is that if w has a form of this equation

$$W = \begin{pmatrix} I & 0 \\ 0 & n^x \end{pmatrix}. \tag{2.20}$$

The second register in Figure 2.9 is initialized to $|1\rangle = |00\ldots01\rangle$, which allows the circuit to produce $|n^x \bmod N\rangle$ at the output. The remaining question is how to implement the

unitary W defined in Equation 2.21. Note that we can express it as follows: $n^x = n^{x_0}.n^{2x_1}.n^{2^2x_2}....n^{2^{t-1}x_{t-1}}$ Thus we can write W as:

$$W = \begin{pmatrix} I & 0 \\ 0 & n \end{pmatrix}^x = \begin{pmatrix} I & 0 \\ 0 & n \end{pmatrix}^{x_0} \begin{pmatrix} I & 0 \\ 0 & n \end{pmatrix}^{2x_1} \cdots \begin{pmatrix} I & 0 \\ 0 & n \end{pmatrix}^{2^{t-1}x_{t-1}},$$

and take

$$V = \begin{pmatrix} I & 0 \\ 0 & n \end{pmatrix}, \tag{2.21}$$

where $V|y\rangle = |ny \mod N\rangle$ and $|y\rangle = |1\rangle$ in our case.
We can write $W = V^{x_0}.(V^{x_1})^2.(V^{x_2})^{2^2}....(V^{x_{t-1}})^{2^{t-1}}$. The gate V are controlled by the qubits of the first register. The figure below represents the circuit that will find the period of the function $f(x) = n^x \mod N$:

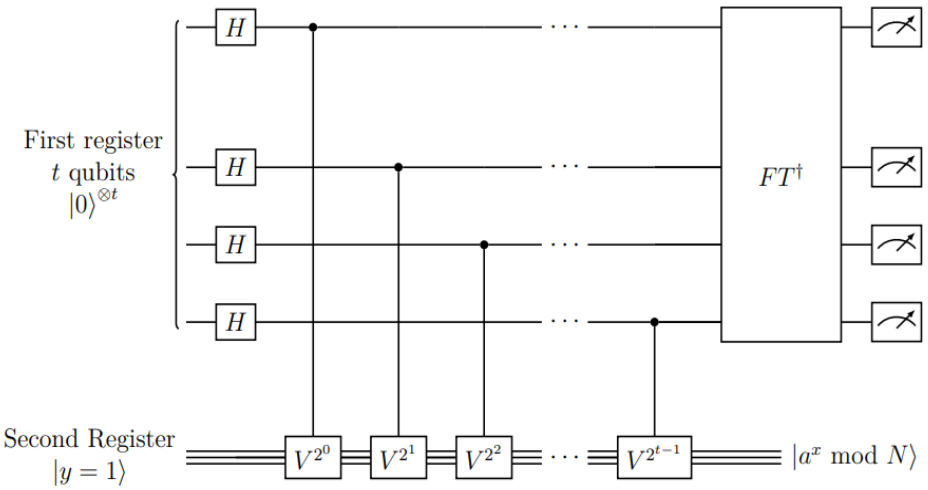


Figure 2.9: Circuit used to determine the period of the function $f(x) = n^x \mod N$.

This circuit is identical to the one used in Quantum Phase Estimation. Consequently, the output yields the eigenvalue of the operator V. The eigenvectors of the unitary operator V take this form:

$$|w_s\rangle = \frac{1}{r}\sum_{k=0}^{r-1} e^{-2\pi i s\frac{k}{r}} \left|n^k\right\rangle, \tag{2.22}$$

with an eigenvalue,

$$\lambda_s = e^{2\pi i\frac{s}{r}}. \tag{2.23}$$

2.5.2 Grover's algorithm

Grover's algorithm [173] performs a search over an unordered set of $N = 2^n$ items for a unique solution. While classical search requires $O(N)$ operations, Grover's algorithm achieves this in $O(\sqrt{N})$ steps, offering a quadratic speedup. It begins with an n-qubit register, all initialized to $|0\rangle$:

$$|0\rangle^{\otimes n} = |0\rangle \,. \tag{2.24}$$

The algorithm begins by preparing the quantum system in a uniform superposition over all possible states. This is accomplished by applying the Hadamard transformation $H^{\otimes n}$, which requires $\mathcal{O}(\log N) = \mathcal{O}(\log 2^n) = \mathcal{O}(n)$ operations, corresponding to n applications of the Hadamard gate.

The following sequence of operations, known as the *Grover iteration*, carries out the amplitude amplification described previously and forms the central component of the algorithm. This iteration is applied $\frac{\pi}{4}\sqrt{2^n}$ times, as originally demonstrated by Grover [71], achieving an optimal probability of measuring the desired state requires an overall phase rotation of $\frac{\pi}{4}$ radians, which occurs on average after $\frac{\pi}{4}\sqrt{2^n}$ iterations.

Each Grover iteration begins with the application of a *quantum oracle*, denoted by $\mathcal{O}$, which conditionally alters the system based on the target configuration. An oracle is essentially a black-box function, and in the quantum setting it can examine and alter the system without causing it to collapse into a classical state. If the system is in the desired state, the oracle applies a phase rotation of π radians; otherwise, it leaves the state unchanged. This operation effectively marks the correct state for subsequent processing. It is worth emphasizing that applying such a phase shift has no impact on the probability of measuring the state, although it negates its amplitude. While quantum oracle implementations often employ an additional ancillary qubit, this implementation does not require one, and the action of the oracle on a basis state $|x\rangle$ can therefore be written as:

$$|x\rangle \xrightarrow{O} (-1)^{f(x)} |x\rangle \,. \tag{2.25}$$

Here, $f(x) = 1$ if x is the correct state and 0 otherwise, depending on the search

problem. The diffusion transform inverts amplitudes of the average, implemented as $H^{\otimes n}$, a conditional phase shift that flips all states except $|x\rangle$, and a final $H^{\otimes n}$. The phase shift is given by $2\,|0\rangle\langle 0| - I$:

$$[2\,|0\rangle\langle 0| - I]\,|0\rangle = 2\,|0\rangle\langle 0|0\rangle - I = |0\rangle\,, \tag{2.26}$$

$$[2\,|0\rangle\langle 0| - I]\,|x\rangle = 2\,|0\rangle\langle 0|x\rangle - I = -\,|x\rangle\,. \tag{2.27}$$

The entire diffusion transform, expressed in terms of $|\psi\rangle$ from Equation 2.27, is given by:

$$H^{\otimes n}[2\,|0\rangle\langle 0| - I]H^{\otimes n} = 2H^{\otimes n}\,|0\rangle\langle 0|\,H^{\otimes n} - I = 2\,|\psi\rangle\langle\psi| - I. \tag{2.28}$$

The Grover iteration can be written as:

$$[2\,|\psi\rangle\langle\psi| - I]O.xx \tag{2.29}$$

Since the cost of the oracle varies with the problem and its implementation, a call to the oracle $\mathcal{O}$ is treated as a single elementary operation when analyzing runtime. One Grover iteration then consists of two Hadamard transformations, contributing $\mathcal{O}(2n)$ operations, together with the application of $\mathcal{O}(n)$ gates required to implement the conditional phase shift [128]. Thus, the overall runtime of a single Grover iteration is $\mathcal{O}(n)$. Since Grover's algorithm performs $\mathcal{O}(\sqrt{N}) = \mathcal{O}(\sqrt{2^n}) = \mathcal{O}(2^{n/2})$ iterations, each with a runtime of $\mathcal{O}(n)$, the total runtime of the algorithm scales as $\mathcal{O}(2^{n/2})$.

Following a sufficient number of Grover iterations, a classical measurement is performed to read out the result. This measurement produces the correct outcome with probability $\mathcal{O}(1)$, thus concluding the algorithm. A summary of Grover's algorithm is provided in [128] as follows:

Input:

- A quantum oracle O where $O\,|x\rangle = (-1)^{f(x)}\,|x\rangle$, and $f(x) = 0$ for all $0 \leq x < 2^n$ except x_0, for which $f(x_0) = 1$.
- n qubits initialized to $|0\rangle$

Output: x_0.

Runtime: $O(\sqrt{2^n})$ operations, with $O(1)$ probability of success.

Procedure:

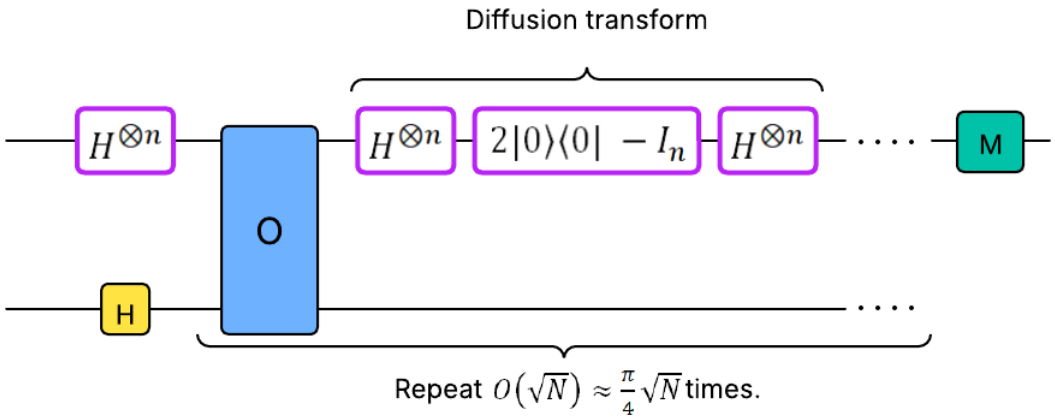


Figure 2.10: Circuit layout for Grover's algorithm

1. $|x\rangle^n$ (initial state)
2. $H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle = |\psi\rangle$
3. $[(2|\psi\rangle\langle\psi| - I)O]^R |\psi\rangle \simeq |x_0\rangle$ (apply the Grover iteration $R \simeq \frac{\pi}{4}\sqrt{2^n}$)
4. x_0 (measure the register)

2.5.3 Quantum Fourier Transform

The Fourier transform [100] is a cornerstone of scientific computing, and its fast implementation, the FFT, underlies many efficient classical algorithms. Likewise, quantum algorithms, including phase estimation and Shor's algorithm, rely on the QFT as a fundamental subroutine, and has motivated related techniques including the fast Fermionic Fourier transform (FFFT). For any j in the computational basis, the QFT is defined by:

$$U_{FT}|j\rangle = \frac{1}{\sqrt{N}} \sum_{k\in[N]} e^{i2\pi\frac{kj}{N}} |k\rangle . \tag{2.30}$$

In particular

$$U_{FT}|0^n\rangle = \frac{1}{\sqrt{N}} \sum_{k\in[N]} |k\rangle H^{\otimes n}|0^n\rangle . \tag{2.31}$$

By representing integers in binary

$$k = (k_{n-1}\dots k_0.), \quad j = (j_{n-1}\dots j_0.), \tag{2.32}$$

we have

$$\begin{aligned}\frac{kj}{N} &= k_0\frac{j}{2^n} + k_1\frac{j}{2^{n-1}} + \ldots + k_{n-1}\frac{j}{2} \\ &= k_0\left(.j_{n-1}\ldots j_0\right) + k_1\left(j_{n-1}.j_{n-2}\ldots j_0\right) + \ldots k_{n-1}\left(j_{n-1}\ldots j_1.j_0\right).\end{aligned}$$

Therefore the exponential can be written as

$$e^{i2\pi\frac{kj}{N}} = e^{i2\pi k_0(.j_{n-1}\ldots j_0)}e^{i2\pi k_1(.j_{n-2}\ldots j_0)}e^{i2\pi k_{n-1}(.j_0)}. \tag{2.33}$$

The key step of the QFT is the following calculation:

$$\begin{aligned}U_{FT}\left|j_{n-1}\ldots j_0\right\rangle &= \frac{1}{\sqrt{2^n}}\sum\nolimits_{k_{n-1\ldots,k_0}} e^{i2\pi k_0(.j_{n-1}\ldots j_0)}e^{i2\pi k_1(.j_{n-2}\ldots j_0)}e^{i2\pi k_{n-1}(.j_0)}\left|k_{n-1\ldots,k_0}\right\rangle \\ &= \frac{1}{\sqrt{2^n}}\left(\sum\nolimits_{k_{n-1}} e^{i2\pi k_{n-1}(.j_0)}\left|k_{n-1}\right\rangle\right)\otimes\left(\sum\nolimits_{k_{n-2}} e^{i2\pi k_{n-2}(.j_1.j_0)}\left|k_{n-2}\right\rangle\right) \\ &\otimes\ldots\otimes\left(\sum\nolimits_{k_0} e^{i2\pi k_0(.j_{n-1}\ldots j_0)}\left|k_0\right\rangle\right) \\ &= \frac{1}{\sqrt{2^n}}\left(\left|0\right\rangle + e^{i2\pi(.j_0)}\left|1\right\rangle\right)\otimes\left(\left|0\right\rangle + e^{i2\pi(.j_1.j_0)}\left|1\right\rangle\right)\otimes\ldots\otimes\left(\left|0\right\rangle + e^{i2\pi(.j_{n-1}\ldots j_0)}\left|1\right\rangle\right).\end{aligned} \tag{2.34}$$

Equation 2.33 involves a sequence of controlled rotations given by

$$\left|0\right\rangle \longrightarrow \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{i2\pi(.j_{n-1}\ldots j_0)}\left|1\right\rangle\right). \tag{2.35}$$

The implementation of QFT circuit is shown below in Figure 2.11.

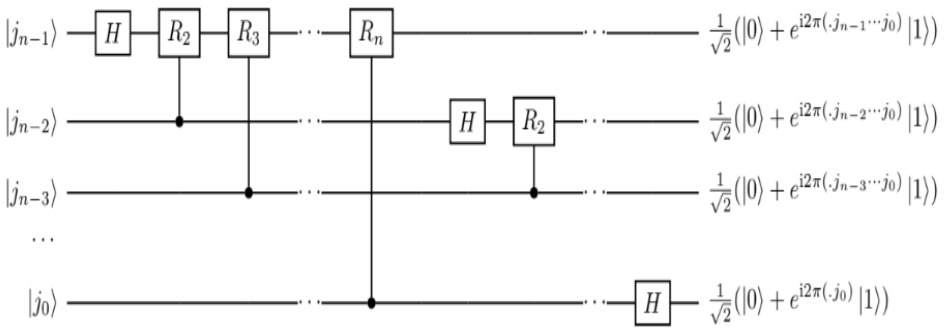


Figure 2.11: Quantum circuit implementing the Quantum Fourier Transform (QFT).

2.5.4 Variational Quantum Eigensolver

VQE is a heuristic strategy [92] for solving combinatorial optimization problems using a hybrid quantum-classical approach. This algorithm is specifically designed to determine the ground state energy E_0 of an n-qubit Hamiltonian, that is, to identify the configuration corresponding to the state $|\psi_0\rangle$ that

$$\hat{H}\,|\psi_0\rangle = E_0\,|\psi_0\rangle\,. \tag{2.36}$$

Even an ideal quantum computer would find this problem difficult. However, it is widely believed that the Hamiltonians describing physical systems rarely correspond to the most difficult instances of the problem. This observation suggests that heuristic quantum algorithms may achieve performance advantages over classical methods. A Hamiltonian of N qubits system is given by:

$$H = \sum_{\alpha} h_\alpha P_\alpha = \sum_{\alpha} h_\alpha \bigotimes_{j=1}^{N} \sigma_{\alpha_j}^{(j)}. \tag{2.37}$$

Here, the coefficients h_α are real numbers, and the operators P_α, known as Pauli strings, are products of Pauli matrices. The VQE algorithm proceeds according to the following steps (see Figure 2.12):

1. Initialize a parameterized trial quantum state $|\psi(\boldsymbol{\theta})\rangle$, specified by a set of variational parameters $\boldsymbol{\theta}$.

2. Estimate the values of the Pauli operators expected to appear in the Hamiltonian by evaluating
$$\langle\psi(\boldsymbol{\theta})|P_\alpha|\psi(\boldsymbol{\theta})\rangle.$$

3. Compute the energy associated with the trial state as
$$E(\boldsymbol{\theta}) = \sum_{\alpha} h_\alpha \langle\psi(\boldsymbol{\theta})|P_\alpha|\psi(\boldsymbol{\theta})\rangle,$$
by aggregating the measurement outcomes obtained in the previous step.

4. Update the variational parameters $\boldsymbol{\theta}$ using the evaluated energy and the results from earlier iterations.

The algorithm proceeds through successive iterations until the ground-state energy has been identified.

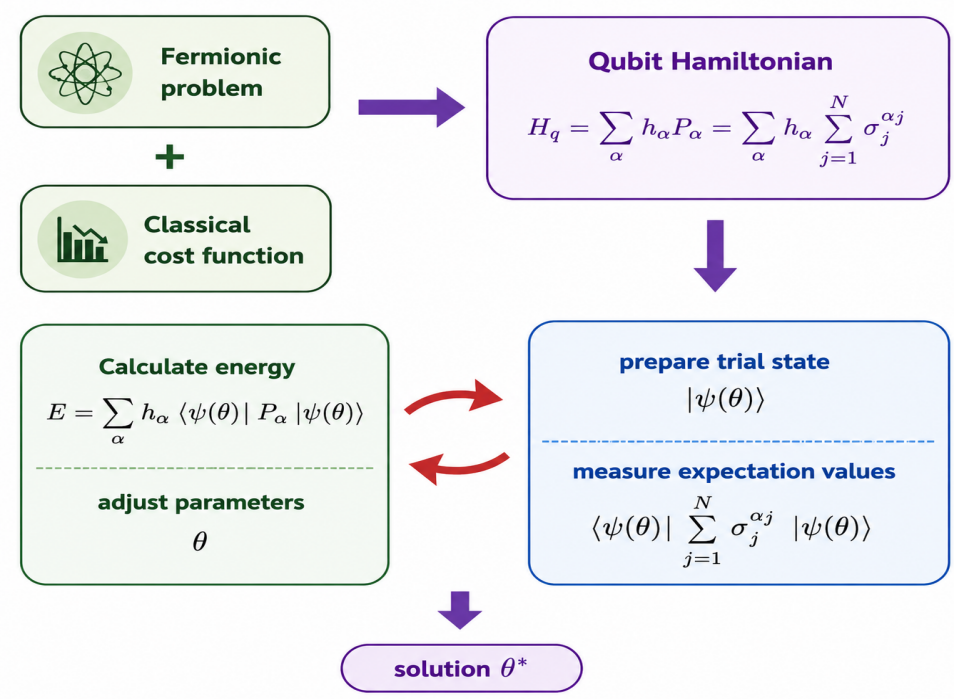


Figure 2.12: Variational quantum eigensolver method.

2.5.5 Quantum machine learning algorithms

Over the past few years, many studies have highlighted the advantages of incorporating quantum methods into learning algorithms [185]. While early work in this area was primarily aimed at reducing computational complexity and achieving algorithmic speedups, more recent studies have investigated how quantum approaches can offer alternative learning representations. These representations may yield solutions distinct from those produced by classical machine learning methods and, in many cases, demonstrate superior performance. Figure 2.13 presents the timeline progression of QML milestones, showing the key developments that have contributed to the growth of QML over time.

In the next section, we will give more details about QML.

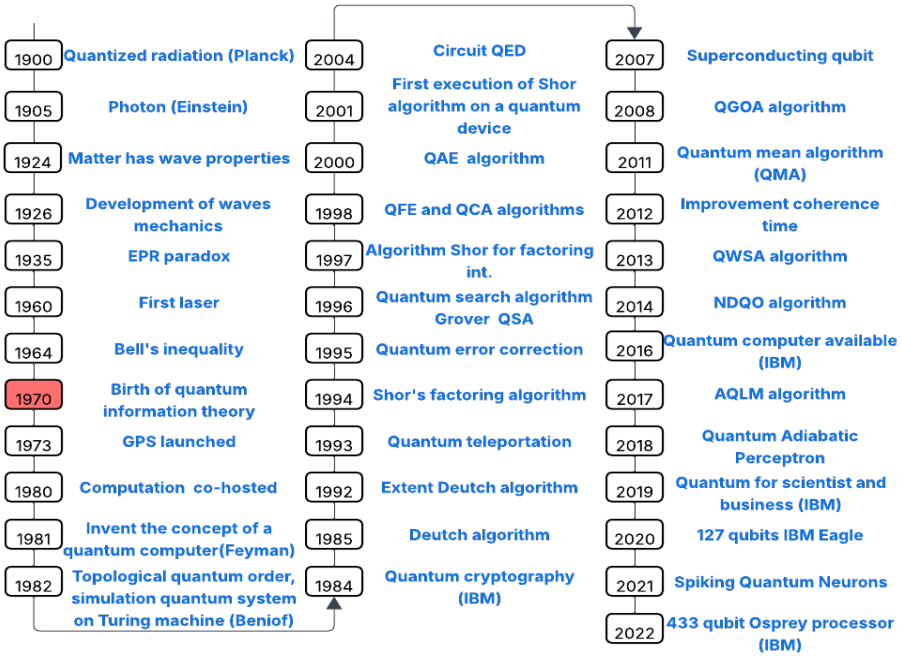


Figure 2.13: Timeline progression of QML milestones.

2.6 Conclusion

This chapter presented the fundamental principles of quantum computing and highlighted its potential as a new computational paradigm capable of addressing complex problems beyond the capabilities of classical computing. The chapter began with the basic concepts of quantum mechanics, including qubits, superposition, entanglement, quantum interference, the observer effect, quantum gates, measurement, and quantum circuits. These concepts form the theoretical foundation of quantum computing and explain how quantum systems can represent and manipulate information in fundamentally different ways from classical systems.

The chapter also examined several important quantum algorithms that demonstrate the computational advantages of quantum computing. Algorithms such as Shor's algorithm and Grover's algorithm illustrated the potential of quantum systems in optimization, factorization, and search problems, while the Quantum Fourier Transform highlighted its central role in many quantum computational processes. In addition, hybrid and variational approaches such as the Variational Quantum Eigensolver (VQE) and

Quantum Machine Learning (QML) algorithms were discussed as promising techniques for solving complex learning and optimization tasks using near-term quantum devices.

Overall, this chapter established the theoretical and computational background necessary for understanding how quantum computing can be integrated with machine learning and NLP. The presented concepts and algorithms provide the essential foundation for the subsequent chapters of this thesis, which focus on the application of Quantum Machine Learning techniques to NLP tasks.

Chapter 3

Quantum Machine Learning

3.1 Introduction

In 1981, Richard Feynman introduced the idea of computation using quantum systems, emphasizing their potential for simulating physical phenomena. This was followed in 1985 by David Deutsch's proposal of a universal quantum computer. In 1994, Peter Shor presented a quantum algorithm for integer factorization, revealing the significant computational advantages of quantum machines. Two years later, Lov Grover proposed an algorithm for efficient search in unstructured databases. Experimental progress soon followed these theoretical advances. In 1998, Jones, Hansen, and Mosca of the University of Oxford successfully implemented Grover's algorithm on a two-qubit quantum computer. In 2001, IBM, in collaboration with Stanford University, demonstrated Shor's algorithm on a seven-qubit quantum processor. In 2012, Preskill introduced the concept of quantum supremacy, describing the point at which controlled quantum systems outperform classical computers in specific tasks. This milestone was reached in 2019, when Google announced the experimental demonstration of quantum supremacy. Although still in its formative stages, QML has already yielded promising algorithms that surpass classical approaches in specific tasks and holds considerable potential to play an increasingly influential role in future advanced machine learning applications. This chapter establishes the theoretical and methodological background necessary for understanding how Quantum Machine Learning models can be applied to complex data-driven tasks, particularly in the context of Natural Language Processing, which represents the central focus of this thesis.

3.2 QML types

The integration of quantum computing and machine learning can be organized into four distinct strategies [203]. These strategies are determined by the nature of the data originating from either a classical (**C**) or a quantum (**Q**) system and by the type of computational platform used, which may be classical (**C**) or quantum (**Q**), as depicted in Figure 3.1.

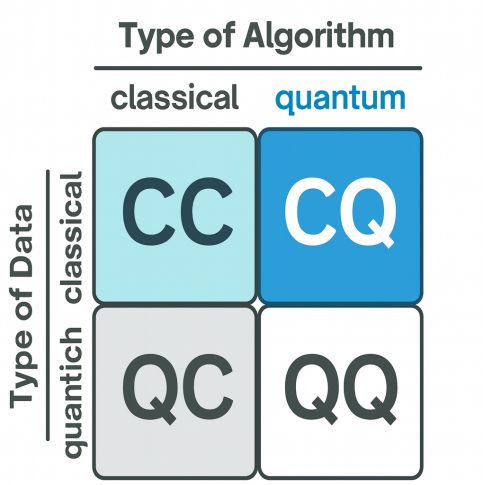


Figure 3.1: Intersection of quantum computing and machine learning.

CC — Classical Dataset processed in Classical Computers. This strategy represents the classical machine learning ML approach where the input of the system is classical data and the algorithm applied is a traditional technique of machine learning .

QC — Quantum Dataset processed in Classical Computers. This strategy consists of manipulating the quantum data by a classical computer, for example: Classifying quantum states emitted by a physical experiment.

CQ — Classical Dataset processed in Quantum Computers. This strategy uses quantum computers to treat classical data. It finds efficient solutions to problems typically solved with ML but on quantum computers.

QQ — Quantum Dataset processed in Quantum Computers.

In this thesis, we focus on the **CQ** scenario. There are two approaches of this strategy:

- The first approach involves executing traditional machine learning algorithms on quantum computers or simulators to obtain potential algorithmic speedups. This approach requires a quantum data-encoding method.
- The second approach involves manipulating classical data using QML algorithms, which will be discussed in detail later.

The Figure 3.2 below, provides a visual comparison between the process of ML and QML. In ML, the input data is directly fed into the algorithm, which operates on it and yields an output. While, QML requires an additional step in which classical data is encoded into quantum data; the algorithm then takes quantum data as input, processes it, and produces quantum data as output. Finally, this quantum data is transformed back into classical data through the measurement process for interpretation.

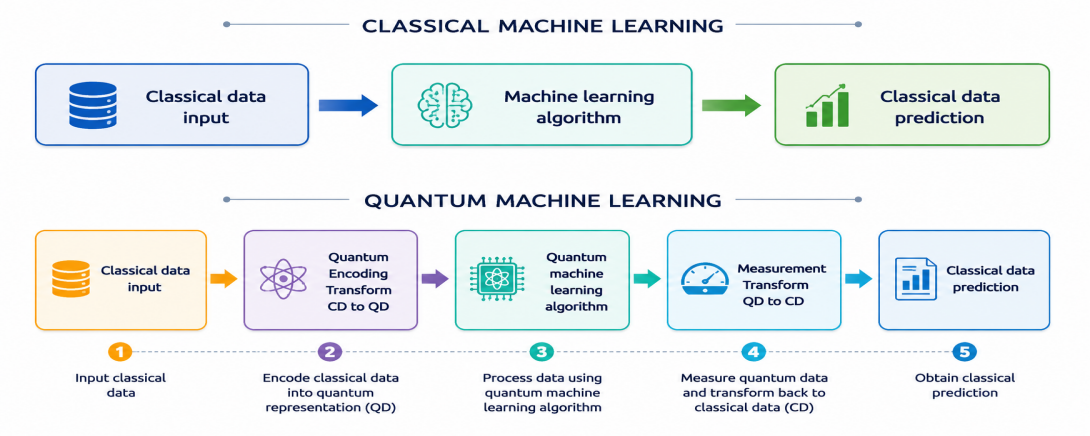


Figure 3.2: ML Vs QML.

3.3 QML models

This section introduces the most employed QML models. They leverage the principles of quantum mechanics and have demonstrated utility in various applications, including data classification. A detailed description of each model and its architecture is provided below.

3.3.1 Quantum Boltzmann machines

The Hamiltonian model that describes QBM is given by $H_\theta = \sum_{i=0}^{p-1} \theta_i \otimes_{j=0}^{n-1} \sigma_{j,i}$ where $\theta \in \mathbb{R}^P$, and $\sigma_{j,i} \in \{I, X, Y, Z\}$ operating on the jth qubit. Analogous to classical Boltzmann machines [1], QBMs are defined using an Ising model, which is a mathematical model used in statistical mechanics and physics to describe interactions between discrete variables. In the network, the nodes, represented by Pauli operators $\sigma_{j,i}$, are defined with respect to specific subsets of qubits. The set of qubits that influence the QBM's output are referred to visible qubits, while the remaining qubits serve as hidden qubits. The following probability :

$$P_v^{QBM} = Tr\left[\Lambda_v \rho^{Gibbs}\right], \tag{3.1}$$

represents the probability of measuring $|v\rangle$ where v is a particular configuration of the visible qubits, it is defined relative to $\Lambda_v = |v\rangle\langle v| \otimes I$ in the quantum Gibbs state [210, 50], which is defined by:

$\rho^{Gibbs} = \frac{e^{-H_\sigma/k_B T}}{Z}$, with $Z = Tr\left[e^{-H_\sigma/k_B T}\right]$ and $k_B T$ is the Boltzmann's constant.

ρ^{Gibbs} represents the probability of a system occupying a specific state based on the system's temperature T and the energy of that state, it is called the Gibbs state.

3.3.2 Variable depth quantum circuits

To overcome the limitations of the VQC model [157], the work [80] proposes automatically adapting the quantum circuit structure using the qBAS metric [18], a benchmark measure for quantum–classical hybrid systems.

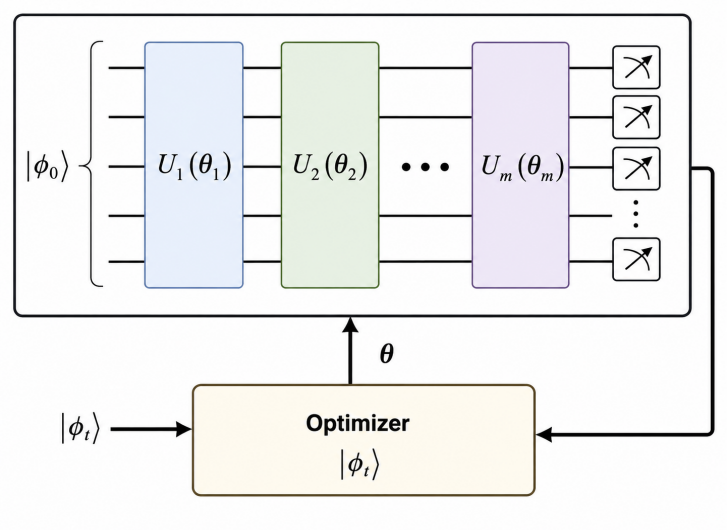


Figure 3.3: VQC framework.

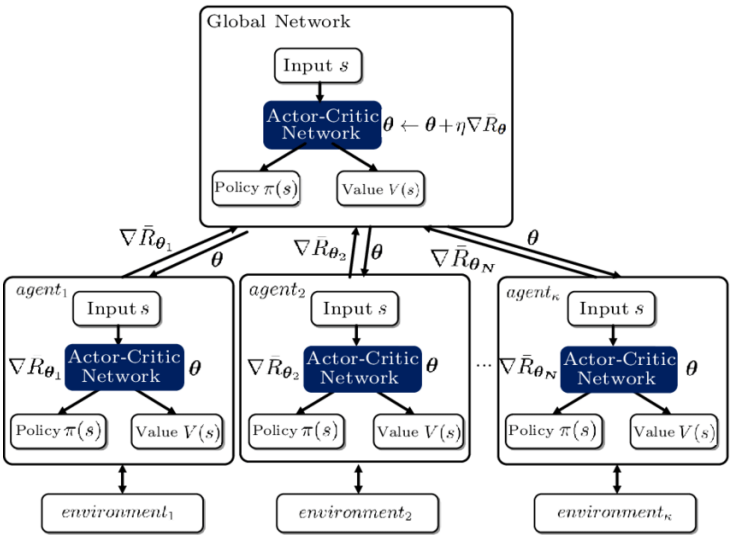


Figure 3.4: A3C learner.

The quantum circuit in Figure 3.3 uses $e^{-i\theta\sigma}$ rotation gates, where $\sigma \in \{X, Y, Z\}$ Pauli gates act at each qubit. And θ represents the angle of the rotation. The framework for asynchronous training of these circuits, as proposed by [35], is depicted in Figure 3.4. The Quantum Asynchronous Advantage Actor-Critic (QA3C) consists of a set of multiple parallel workers and global parameters. Local agents compute gradients and store them in process-specific memory. The gradients are uploaded to a global shared memory to update the global model parameters, which are then immediately downloaded to the contributing

agent.

3.3.3 Hybrid quantum autoencoders

This model is proposed in [172], HQA combines artificial neural networks (ANNs), with quantum neural networks (QNNs) constructed from parameterized quantum circuits (PQCs). The model uses an encoder–decoder architecture, the encoder maps a quantum state from the Hilbert space $H_2^{\otimes n}$ to a subspace of the real vector space V with dimension $v = \dim(V)$, the decoder implements the inverse operation. As shown in Figure 3.5, the encoder ϵ is implemented as a quantum circuit parameterized by the vector α, and the circuit takes an input state $|\psi_{\text{in}}\rangle$ then applies the unitary $U_1(\alpha)$ to the overall system.

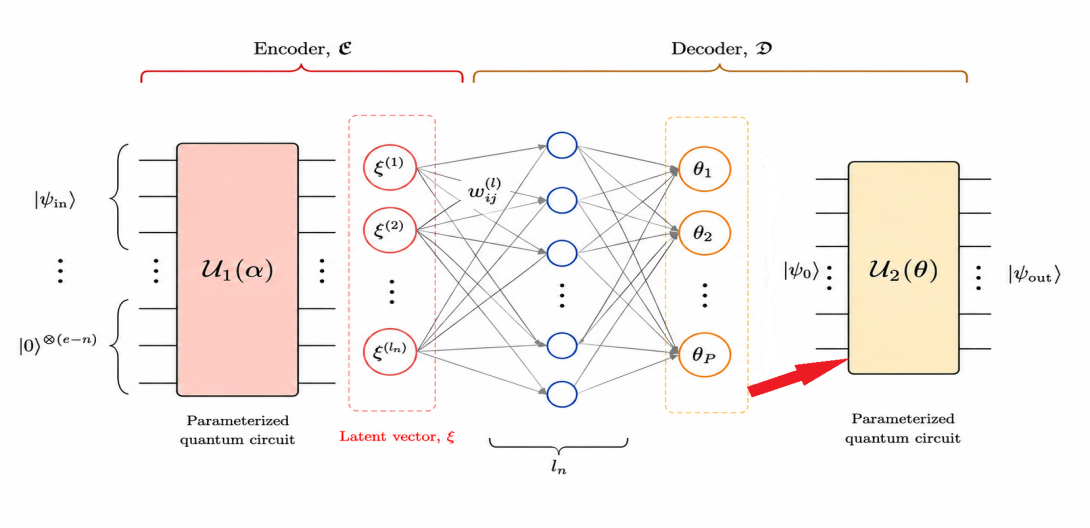


Figure 3.5: HQA model.

3.3.4 Quantum reservoir computing

This model exploits a dynamic quantum system for temporal information processing [176]. The goal is to generate a target output sequence given an input sequence,the system dynamics in the QRC framework are expressed as:

$$\rho_t = T_{u_t}\left(\rho_t - 1\right). \tag{3.2}$$

In this expression, ρ_t represents the system's density operator at instant time t, while T_{u_t} is an input-dependent, completely positive, trace-preserving CPTP (see Equation 3.3) map governing the evolution of the quantum reservoir.

$$\rho_t = e^{-iH\tau} \left(\rho_{input} \otimes Tr_{input} \left(\rho_t - 1 \right) \right) e^{iH\tau}, \tag{3.3}$$

where $\rho_{input} = |\psi_{u_t}\rangle \langle \psi_{u_t}|$ with $\psi_{u_t} = \sqrt{1-u_t}\,|0\rangle \langle 0| + \sqrt{u_t}\,|1\rangle \langle 1|$. An auxiliary qubit is initialized with an input value $u_t \in [0,1]$, after which the entire quantum reservoir (QR) system undergoes time evolution governed by $e^{-iH\tau}$ which is the input-independent unitary operator .

3.3.5 Support vector machines with a quantum kernel estimator

QSVM and QSVM-Kernel employs the quantum state space as its feature space. to efficiently evaluate kernel values. Classical input data $\overrightarrow{x}$ is mapped nonlinearly onto an N-qubit quantum state via the circuit $\Gamma_{\phi(\overrightarrow{x})}$ applied to the initial state $\left|0^{\otimes N}\right\rangle$:

$$|\phi\left(\overrightarrow{x}\right)\rangle = \Gamma_{\phi\left(\overrightarrow{x}\right)} \left|0^{\otimes N}\right\rangle. \tag{3.4}$$

The mapping $\Gamma_{\phi\left(\overrightarrow{x}\right)}$ projects the data into a feature space of dimension 2^N where N represents the number of qubits.

$$\Gamma_{\phi\left(\overrightarrow{x}\right)} = U_{\phi\left(\overrightarrow{x}\right)} H^{\otimes N} U_{\phi\left(\overrightarrow{x}\right)} H^{\otimes N}. \tag{3.5}$$

In this circuit, $U_{\phi(\overrightarrow{x})}$ encodes the classical input data (Figure 3.6), and H represents a Hadamard gate. In the training phase, kernel entries are derived from the training data to construct a separating hyperplane. In testing, the kernel values between new inputs $\overrightarrow{x}$ and the training data are evaluated, allowing classification according to the learned hyperplane.

In QSVMs, kernel evaluation is performed on a quantum computer, while hyperplane optimization and data classification are carried out classically, as in standard SVMs. Projected quantum kernels were introduced in [112] and demonstrated superior performance over all tested classical models. Their approach first applies PCA [85] to convert each image into an n-dimensional vector, which is then embedded into a quantum circuit [157, 156, 169], or a Hamiltonian circuit , or an IQP circuit [74].

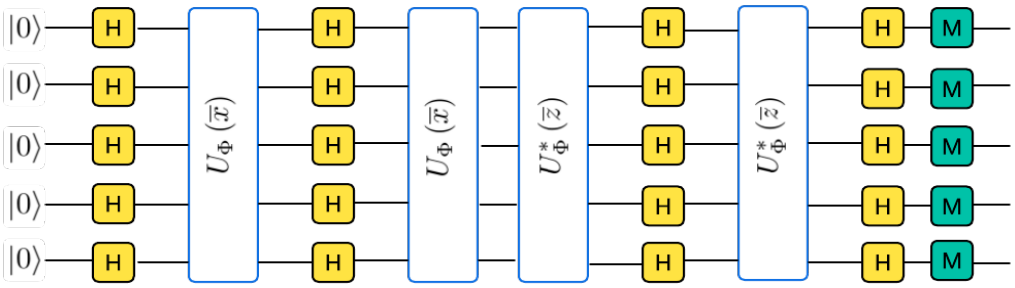


Figure 3.6: Quantum circuit of SVM.

3.3.6 Quantum neural networks

QNN consist of quantum neurons, whose model is detailed in [33]. Each neuron takes a quantum state $|x_i\rangle$ as input and produces a real-valued output o (Figure 3.7), with the quantum weight $R(\theta_i)$ implemented as a rotation gate. $R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$, Controlled-NOT gate $U(\tau) = C(f(\tau))$ where f is the sigmoid function, and the function $C(x)$ is $C(k)|\phi\rangle = \begin{bmatrix} \cos\left(\frac{\pi}{2}m - \alpha\right) \\ \sin\left(\frac{\pi}{2}m - \alpha\right) \end{bmatrix}$. Expanding on the quantum neuron model, QNNs are introduced in [180, 15]. Classical input and weight vectors of size m are encoded on quantum hardware using $N = \log_2 m$ qubits, with data represented as relative quantum phases (± 1) in superpositions.

QNN training can follow two strategies: global variational training, which inverts a quantum state $|\psi_w\rangle$ back to $|1\rangle^{\otimes N}$, and local variational training, where $V(\vec{\theta})$ is decomposed into layers $V_j(\vec{\theta_j})$ trained sequentially via a cost function.

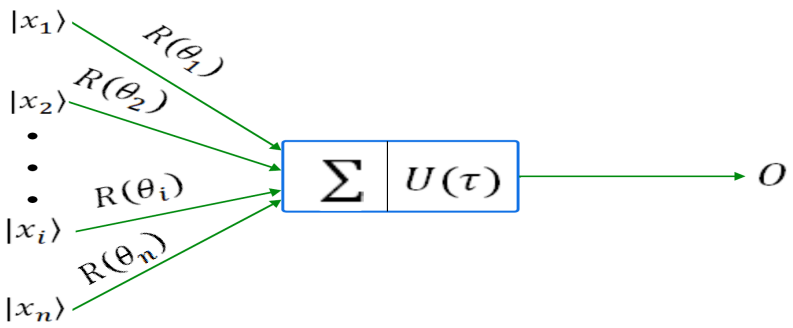


Figure 3.7: Quantum neuron model.

3.3.7 Hybrid k-neighbors-nearby model

In quantum computing, the similarity measure is the overlap between two states functions, analogous to Euclidean distance, and can be evaluated using the swap-test circuit (Figure 3.8). This technique consists of computing distances between classical vectors in K-

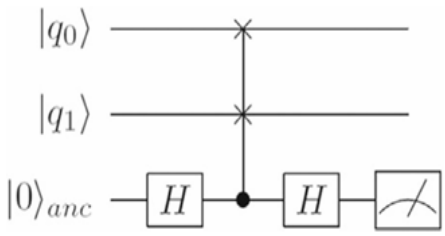


Figure 3.8: Test swap circuit.

Nearest Neighbors (KNN) algorithms [94]. The input is encoded as quantum states $|0\rangle$ or $|1\rangle$, in the form $|q_1\rangle \dots |q_n\rangle$. Controlled-SWAP (Fredkin) gates, driven by an auxiliary qubit, are then applied.

The reference state is processed via a unitary V (Figure 3.9) to form the test circuit. Comparing each qubit of the reference and test states, if $|\psi\rangle = |\phi\rangle$, the auxiliary qubit is measured in $|0\rangle$ with probability one. For multiple qubits, the probability is given by:

$$P\left(|0\rangle_{anc}\right) = \frac{1}{2^n} + \frac{1}{2^n}\sum_{i=1}^{n} \left|\langle\psi_i|\phi_i\rangle\right|^2 . \tag{3.6}$$

In this procedure, the ith qubits of the two quantum states are combined, and n denotes the number of qubits appended to the test circuit. The probability obtained upon measurement serves as a measure of the overlap between the states. By evaluating d distinct input–reference state pairs, one obtains a classical vector $S \in \mathbb{R}^d$, where each component represents the measured overlap probability, such that:

$S = [P_1, P_2, \dots P_d]$.

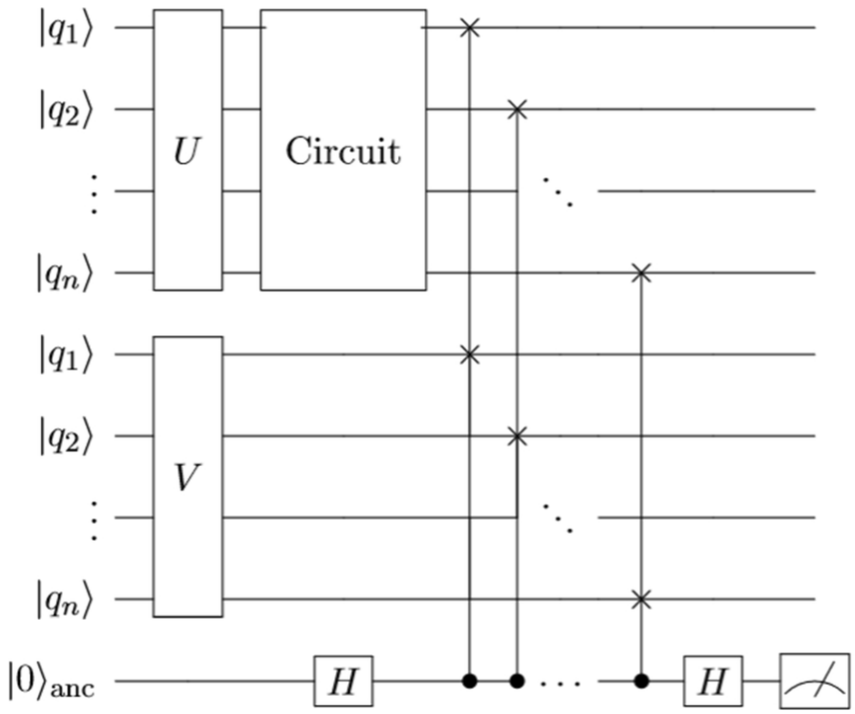


Figure 3.9: QKNN circuit.

3.3.8 Orthogonal neural networks

Orthogonal neural networks (OrthoNNs) are a new class of networks that enforce orthogonal weight matrices [84, 193]. Classical methods for maintaining orthogonality during gradient descent are often slow and only approximate. Typically, OrthoNNs use constraints like the Stiefel manifold [84], with the key challenge being exact orthogonality during training.

To address this, [88] proposes the Pyramidal Circuit, a neural network layer that performs orthogonal matrix multiplication, allowing gradient updates while perfectly preserving orthogonality at standard layer execution times.

The quantum implementation realizes an orthogonal weight matrix with a fully connected layer using only Reconfigurable Beam Splitter (RBS) gates. This gate is parame-

terized by an angle $\theta \in [0, 2\pi]$, with the following matrix:

$$RBS\left(\theta\right) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad RBS\left(\theta\right) : \begin{cases} |01\rangle \mapsto \cos\theta\,|01\rangle - \sin\theta\,|10\rangle \\ |10\rangle \mapsto \sin\theta\,|01\rangle + \cos\theta\,|10\rangle \end{cases} \tag{3.7}$$

Consider two qubits, one in $|0\rangle$ and the other in $|1\rangle$. The RBS gate performs a swap operation in superposition: the qubit in state $|0\rangle$ remains unchanged with amplitude $\cos\theta$, or it is swapped with amplitude $-\sin\theta$ when the new thread is above ($|01\rangle \mapsto |10\rangle$), or with amplitude $+\sin\theta$ otherwise, if below, see (Figure 3.10). Implementation can use either a single FSIM gate or a sequence of four Hadamard gates, two R_y rotations, and two controlled-Z gates (Figure 3.11).

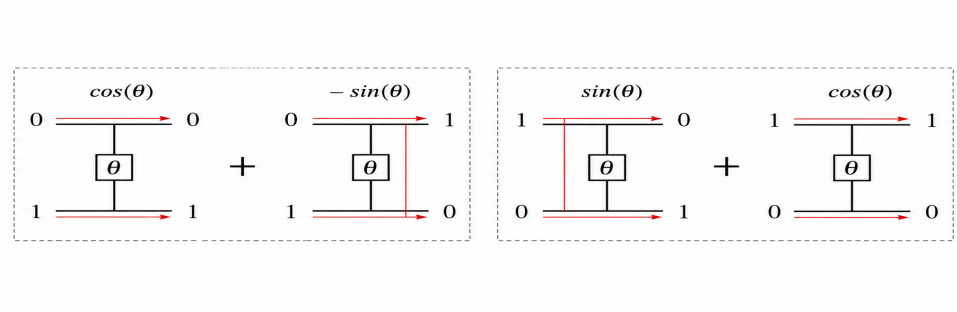


Figure 3.10: Quantum mapping in two qubits.

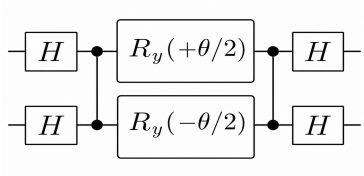


Figure 3.11: Decomposition of the RBS gate.

3.3.9 Quantum fully self-supervised neural networks

The work of [93] introduces a quantum neural network using qudits. A qudit refers to a D-level quantum system, with ($D > 2$), where a general state is a superposition of these

basis states $|0\rangle, |1\rangle, \ldots, |D-1\rangle$:

$$|\psi\rangle = b_0 |0\rangle + b_1 |1\rangle + \ldots + b_{D-1} |D-1\rangle = \begin{bmatrix} b_0 \\ \cdots \\ \cdots \\ b_{D-1} \end{bmatrix}, \tag{3.8}$$

with this normalization criterion $|b_0|^2 + |b_1|^2 + \ldots + |b_{D-1}|^2 = 1$. In this qudit-based QNN model, classical inputs x_k are represented as D-dimensional vectors. The network function $f_{\mathrm{QNNM}}(x_k)$ generates a binary output $[0, 1]$ for each input. Where a sigmoid function is applied to quantum states in the range $\left[0, \frac{2\pi}{D}\right]$.

$$f_{QNNM}(x_k) = \frac{1}{1 + e^{-x_k}}. \tag{3.9}$$

Next, assign this result to the amplitude corresponding to the kth ground state.

$$|b_k\rangle = \left(\frac{2\pi}{D} f_{QNNM}(x_k)\right). \tag{3.10}$$

The QFS-Net architecture [93] exemplifies such a network, using qutrits as the core information-processing units. Within each layer, qutrit neurons are implemented using a transformation gate T, while the interconnection weights are encoded through Hadamard gates H. The rotation angle, illustrated by the pink arrow in Figure 3.12, is determined by the relative information difference between a qutrit neuron and its neighboring neurons, and is subsequently applied in the rotation gate to update the inter-layer connections.

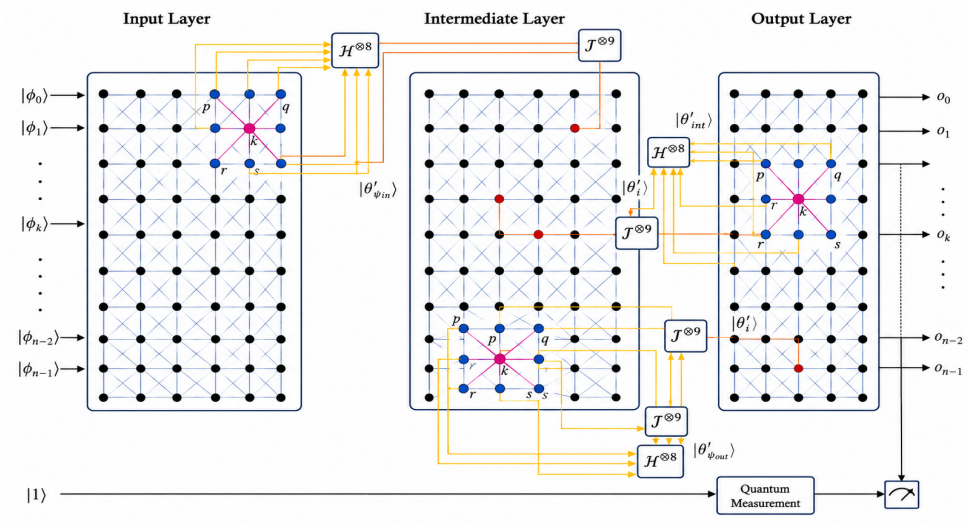


Figure 3.12: QFS-Net.

3.3.10 CNOT neural networks

QNN often requires complex unitary operations or many auxiliary qubits to implement activation functions. To overcome these limitations, the CNOT Measured Network (CMN) was proposed in [103]. This architecture uses only CNOT gates and measurement operators, yet can realize universal logic operations like AND and OR while keeping the number of auxiliary qubits constant and maintaining an efficient learning rate.

The model focuses on discrete Boolean functions, using multiple layers with simple transfer functions to reduce computational complexity. Network construction involves initializing an arbitrary quantum state, applying CNOT gates, and adding measurements. Data qubits x_1, x_2 are initialized to reflect the inputs, e.g., $|x_1\rangle = |0\rangle$, while weight qubits are initialized randomly as $w_i = \frac{1}{\sqrt{3}}|0\rangle + \frac{\sqrt{2}}{\sqrt{3}}|1\rangle$ (Figure 3.13).

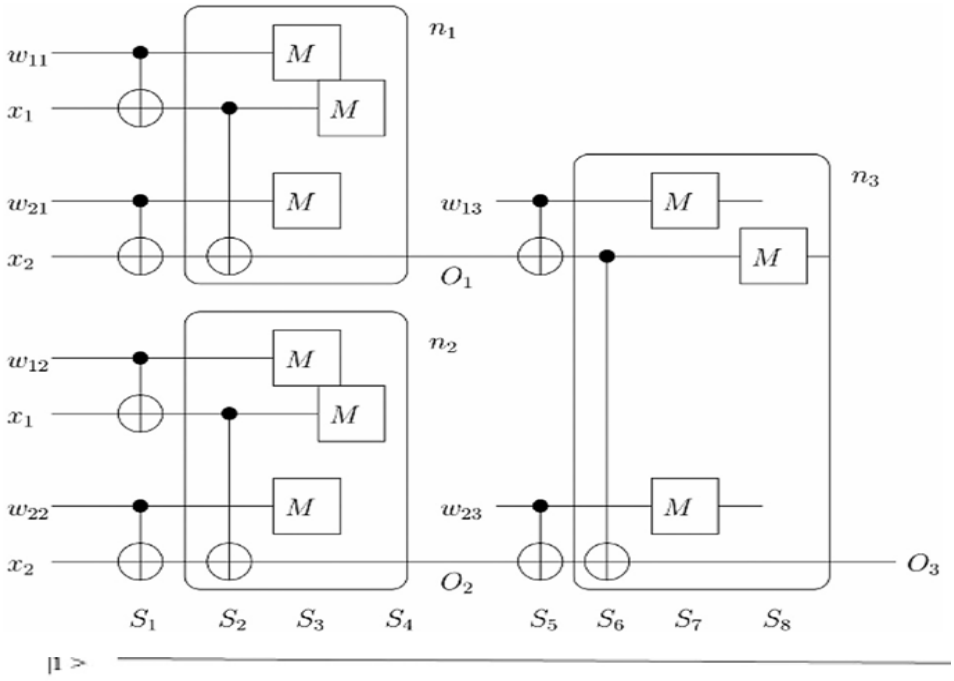


Figure 3.13: CNOT quantum neural network

Data and weight qubits are then interleaved using CNOT gates:

$$|\phi\rangle = CNOT\,|w_{ji} \otimes x_i\rangle = \alpha_{w_{ji}}\alpha_{x_i}\,|0\rangle_{w_{ji}} \otimes |0\rangle_{x_i} \pm \beta_{w_{ji}}\beta_{x_i}\,|1\rangle_{w_{ji}} \otimes |1\rangle_{x_i}\,. \tag{3.11}$$

The network output can be expressed as a reduced density operator using the density

matrix formalism, given by

$$\rho = \sum_i p_i \left|\psi_i\right\rangle \left\langle\psi_i\right| . \tag{3.12}$$

For indices i, the measurement of the qubit state $|x_0 w_0 \ldots x_n w_n\rangle$ is expressed as:

$$\rho^0 = \sum_i p_i tr\left(\left|\psi_i\right\rangle \left\langle\psi_i\right|\right) \otimes \sum_j p_j \left|o_j\right\rangle \left\langle o_j\right| . \qquad \rho^0 = \sum_j p_j \left|o_j\right\rangle \left\langle o_j\right| . \tag{3.13}$$

3.3.11 Quantum deep convolutional neural networks

Quantum adaptations of convolutional networks form another important class of algorithms. In [98], a QDCNN is proposed, where the architecture is inspired by classical CNN, comprising multiple quantum convolutional layers followed by a quantum classifier.

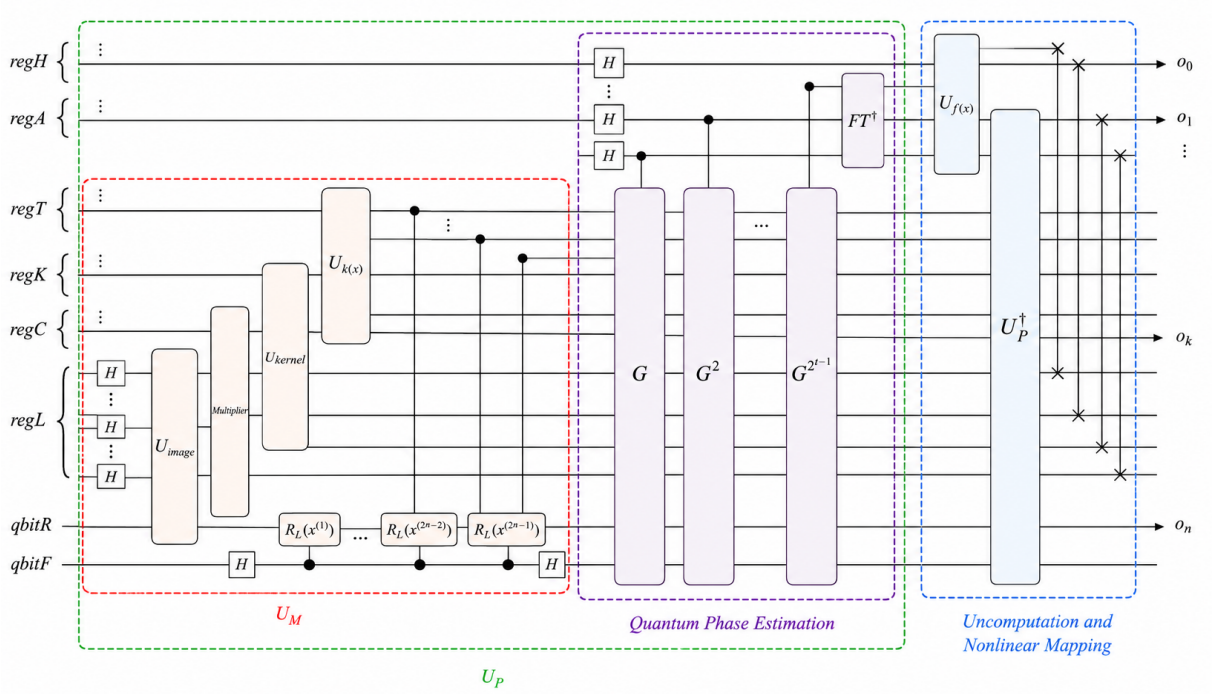


Figure 3.14: QDCNN-net architecture

Figure 3.14 shows the principle of this algorithm where, all qubits are first initialized to $|0\rangle$. Hadamard gates are then applied to the $regL$ register, and the Quantum Random Access Memory (QRAM) algorithm is used to load the input data and convolutional kernel into the $regC$ and $regK$ registers. Next, The registers are next combined via quantum multiplication, and a bifurcation qubit $qbitF$ with a rotation qubit $qbitR$ is added.

$$U_{QRAM} = d_j \left|j\right\rangle \left|0\right\rangle \mapsto \left|j\right\rangle \left|d_j\right\rangle \tag{3.14}$$

3.3.12 Quantum backpropagation neural networks

This model [33] is constructed as shown in Figure 3.15.

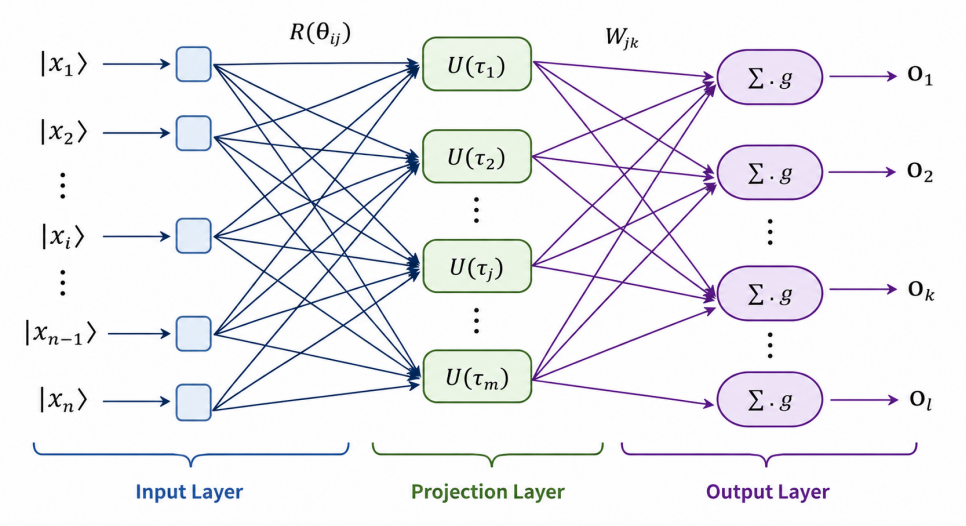


Figure 3.15: QBP architecture.

In this network, the output layer is classical, while both the input and hidden layers consist of quantum neurons. The input vector is defined as

$$X = [|x_1\rangle, |x_2\rangle, \ldots, |x_n\rangle]^T,$$

and the network's output vector is

$$O = [o_1, o_2, \ldots, o_k, \ldots, o_l]^T.$$

The weight matrix connecting the input and hidden layers is

$$V = (v_{i,j})_{n \times m},$$

where $v_{i,j}$ is the weight between the ith input neuron and the jth hidden neuron. As the training samples are real-valued, a specific encoding scheme is required to map them onto quantum states. Given a vector:

$$X = [x_1, x_2, \ldots, x_n]^T,$$

the corresponding quantum input vector is obtained as

$$[|x_1\rangle, |x_2\rangle, \ldots, |x_n\rangle]^T,$$

where each $|x_i\rangle$ represents the quantum encoding of the real value x_i.

$$X = \begin{bmatrix} \cos\left(2\pi \cdot sigmoid\left(x_i\right)\right) \\ \sin\left(2\pi \cdot sigmoid\left(x_i\right)\right) \end{bmatrix}. \tag{3.15}$$

To overcome the problem of local minima in gradient descent, a genetic algorithm (GA) can be employed. Compared to classical GA, the Quantum Genetic Algorithm (QGA) [127] leverages quantum superposition to maintain population diversity, simplifies computations, and reduces operational steps through the use of quantum rotation gates. Combining the QGA with the BP algorithm minimizes the QBP training error and helps avoid convergence to local optima. In this model, chromosomes are represented by qubits as follows:

$$q_j^t = \begin{bmatrix} \alpha_{11}^t & \dots & \alpha_{1k}^t & \alpha_{21}^t & \dots & \alpha_{2k}^t & \dots & \alpha_{m1}^t & \dots & \alpha_{mk}^t \\ \beta_{11}^t & \dots & \beta_{1k}^t & \beta_{21}^t & \dots & \beta_{2k}^t & \dots & \beta_{m1}^t & \dots & \beta_{mk}^t \end{bmatrix}, \tag{3.16}$$

where q_j^t denotes the jth chromosome in the tth generation, k and m represent the number of qubits, and the total number of genes respectively.

3.3.13 Long short-term memory neural networks

This model is employed to capture both short- and long-term dependencies in sequential data.

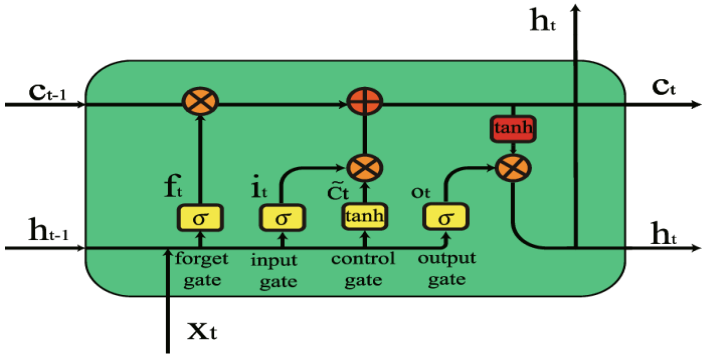


Figure 3.16: LSTM cell.

A typical LSTM cell (Figure 3.16) receives three inputs: the current input x_t, where t indicates the time step corresponding to the unrolled sequence; the previous hidden state h_{t-1}, representing the hidden state from the preceding time step; and the previous cell

state c_{t-1}. Each LSTM cell also contains three gating mechanisms: the input gate (i), the output gate (o) and the forget gate (f). These gates utilize either the sigmoid (σ) or the hyperbolic tangent (tanh) activation functions to regulate the flow of information [31]. A hybrid architecture is proposed in Figure 3.17 by Hong [79]. The architecture comprises two LSTM layers, each containing 32 cells, followed by two fully connected layers. The second fully connected layer consists of 10 neurons, which serve as the interface with QNN. The quantum layer is constructed using an IQP ansatz [74] together with Strongly Entangling Layers [157], and is followed by a final classical output layer.

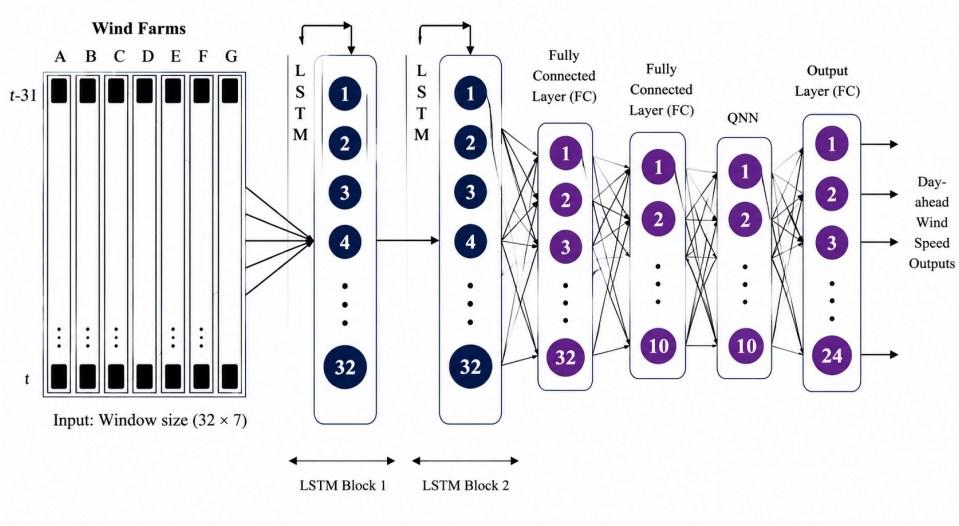


Figure 3.17: LSTM architecture.

Due to the constraint imposed by the 10-qubit quantum layer, the 32-cell LSTM output must first be reduced through a classical dense layer of 10 neurons before being passed to the QNN. After the quantum layer processes the 10-dimensional input, a fully connected classical layer is added to produce the final predictions.

In [194], a meta-learning quantum approximate optimization algorithm (MetaQAOA) is introduced to solve the MaxCut problem [59]. The approach combines a QNN, implemented as a parameterized quantum circuit, with a classical LSTM that functions as an optimizer. This hybrid optimizer accelerates the search for near-optimal QAOA parameters [65]. The algorithm proceeds through the following main steps. First, a uniformly distributed quantum superposition is prepared as the initial state: $|\psi_0\rangle$:

$$|\psi_0\rangle = |+_1\rangle\,|+_2\rangle \ldots |+_n\rangle\,, \tag{3.17}$$

where n denotes the number of nodes in the graph G for the MaxCut problem, and $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ represents a superposition state. The cost Hamiltonian $\widehat{H}_{c1}$ and the mixing Hamiltonian $\widehat{H}_m$ are defined as:

$$\widehat{H}_{c1} = \sum_{(j,k)\in E} \frac{1}{2}\left(\hat{I} - \hat{Z}_j\hat{Z}_k\right), \tag{3.18}$$

$$\widehat{H}_m = \sum_{j\in n} \widehat{X}_j, \tag{3.19}$$

where $\widehat{H}_m$ is the Pauli-sum Hamiltonian consisting of the operator $\widehat{X}$ acting on each qubit, $\widehat{H}_{c1}$ is diagonal in the computational basis, and I, X, Z denote the Pauli operators I, X, and Z, respectively. The indices j and k correspond to the two nodes connected by an edge in the graph G. Third, the unitary operator $\hat{U}(\eta, \gamma)$ applied to this initial state is given by:

$$\hat{U}(\eta, \gamma) = \Pi_{q=1}^{p} e^{-i\eta_q \widehat{H}_m} e^{-i\gamma_q \widehat{H}_{c2}}, \tag{3.20}$$

the number of repetitions of the unitary transformation is denoted by P and the parameters η, γ are defined by:

$$\overrightarrow{\eta} = [\eta_1, \eta_2 \ldots \eta_p]^T \qquad \overrightarrow{\gamma} = [\gamma_1, \gamma_2 \ldots \gamma_p]^T. \tag{3.21}$$

The resulting parameterized circuit is shown in Figure 3.18.

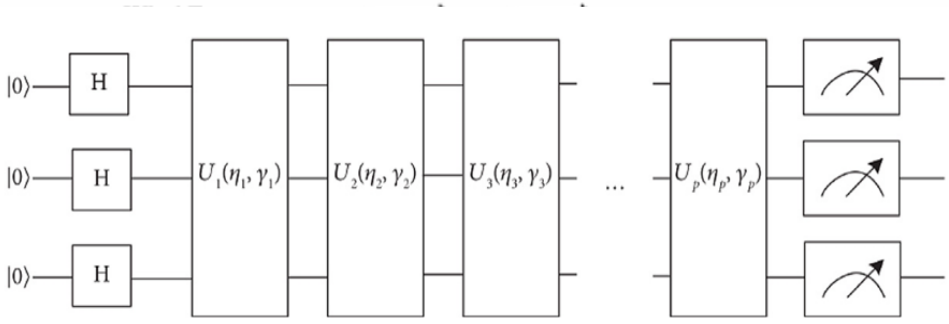


Figure 3.18: LSTM-QNN parameterized circuit.

3.3.14 Feed foward neural networks

This model is detailed in [180, 179]. A quantum FFNN consists of n_i artificial neurons arranged across L_j successive layers. Information is transmitted from the input layer to

the output layer through the interconnections between neurons. Each artificial neuron computes a scalar product between the input vector $\overrightarrow{i}$ and the corresponding weight vector $\overrightarrow{w}$, generating an activation based on $\overrightarrow{i} \cdot \overrightarrow{w}$. Assuming the quantum register of N qubits is initially in the ground state $|0\rangle^{\otimes N}$, the input vector is encoded into the quantum state via a unitary operation V_i, such that

$$|\psi_i\rangle = V_i \, |0\rangle^{\otimes N} . \tag{3.22}$$

Subsequently, the weight vector $\overrightarrow{w}$ is incorporated into the input state by means of an additional unitary transformation V_w, with:

$$|1\rangle^{\otimes N} = V_w \, |\psi_w\rangle . \tag{3.23}$$

By running several quantum register instances in parallel, measurement outcomes can be used to propagate information about the input-weight processing to subsequent layers. Figure 3.19 shows an abstract representation of this architecture.

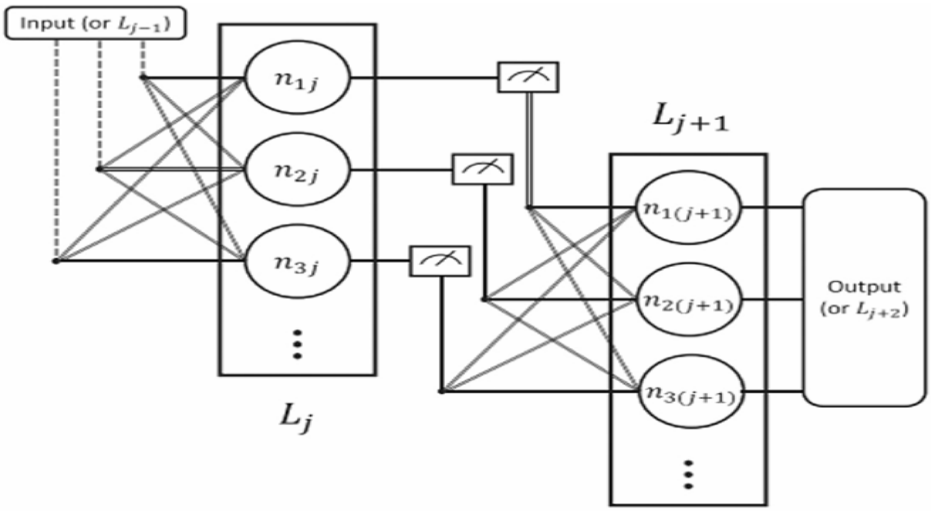


Figure 3.19: Feed Foward Neural Network (FFNN)

3.3.15 Quantum generative adversarial networks

In [6, 102, 167], quantum extensions of the GAN framework [67] are proposed. The discriminator receives as input pairs of samples drawn from the classical training dataset, where y represents the conditional variable. The generator encodes both the conditional

information and the training data distribution into a quantum state $|y\rangle$. The classical discriminator is then trained using both the original training samples and the data produced by the quantum generator, and its output is fed back to update the generator, as illustrated in Figure 3.20.

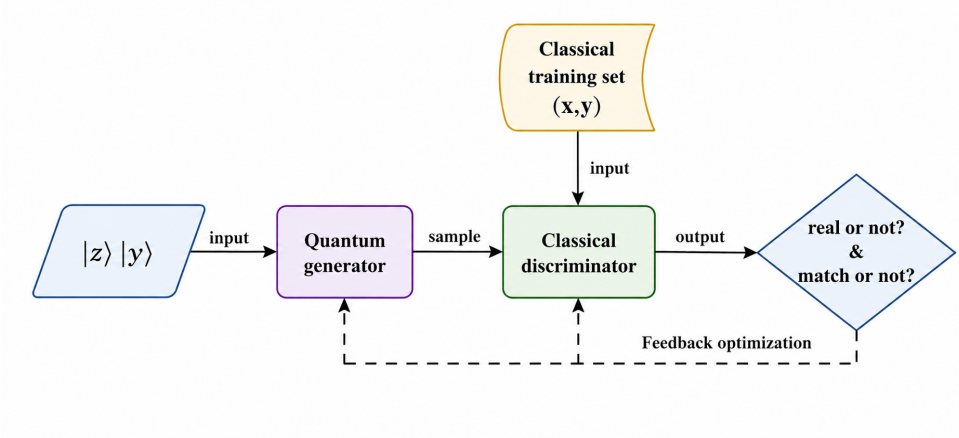


Figure 3.20: Conditional adversarial neural networks.

At the outset, the parameters of both the quantum circuit and the binary classification network are initialized randomly. During the training process, the parameters of the generator and discriminator are updated in an alternating manner. The discriminator assesses both the genuine training data and the samples produced by the quantum model. Through repeated executions of the quantum circuit, the generator incrementally improves its ability to approximate the true data distribution. This iterative optimization procedure continues until the generator accurately reproduces the distribution of the training dataset.

$$|y\rangle = \sum_{j\in n}^{m} \frac{1}{\alpha_j} |y_j\rangle \,. \tag{3.24}$$

In the design of the quantum generator circuit, a parameterized quantum circuit (PQC) is constructed by combining single-qubit rotation gates with multi-qubit gates that induce entanglement among the qubits. The single-qubit rotation gates R_x and R_z are defined as follows:

$$R_x(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}, \; R_z(\theta) = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}. \tag{3.25}$$

The XX operation in Figure 3.21 denotes a two-qubit interaction, where one qubit serves as the control and the other as the target.

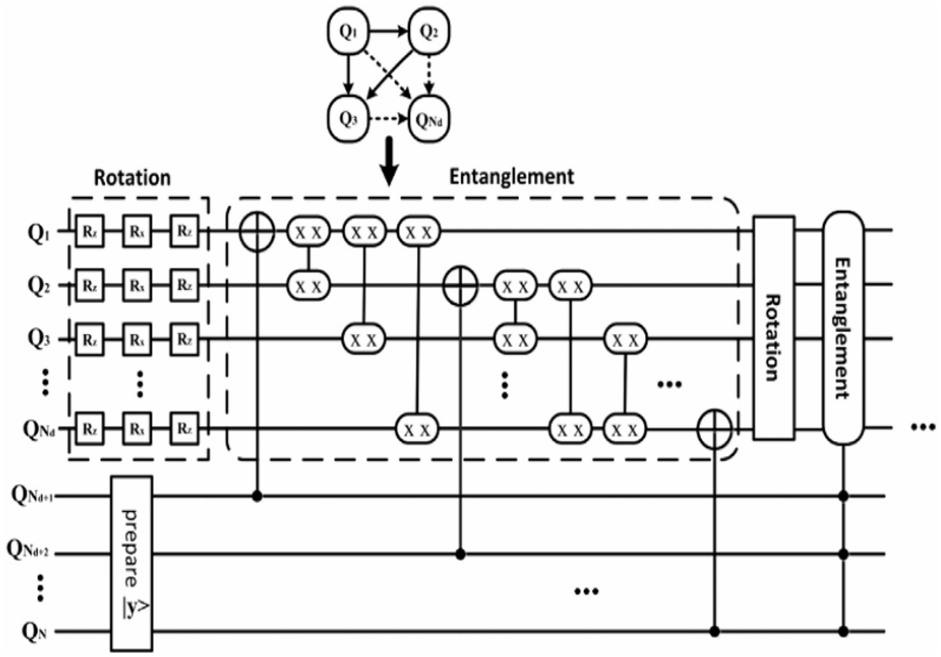


Figure 3.21: QGAN quantum circuit generator.

3.3.16 Quantum Monte Carlo

The work [168], represents the first application of this model in economic problems. It starts by performing Gaussian sampling and subsequently carried out bank stress testing and solved Dynamic Stochastic General Equilibrium (DSGE) models using deep learning techniques. To construct a quantum counterpart of the Monte Carlo algorithm, they introduced two unitary operators [168], A for modeling function $p(.)$ and unitary R for modeling function $f(.)$ in this equation.

$$\mu := E\left[f\left(x\right)\right] = \int_{R^d} p\left(x\right) f\left(x\right) dx. \tag{3.26}$$

First, we define A by considering an m-qubit register initialized in the state $|0\rangle^{\otimes m}$.

$$A\left|0\right\rangle^{\otimes m} = \sum_{i\in x}^{m} \sqrt{p\left(i\right)}\left|i\right\rangle. \tag{3.27}$$

Next, the operator R is introduced, along with an additional ancilla qubit.

$$R\left|i\right\rangle\left|0\right\rangle = \left|i\right\rangle\left(\sqrt{1-f\left(i\right)}\left|0\right\rangle + \sqrt{f\left(i\right)}\left|1\right\rangle\right). \tag{3.28}$$

The final output state is:

$$\left|\chi\right\rangle := F\left|0\right\rangle^{\otimes m+1} = \sum_{i\in x}^{m}\sqrt{p\left(i\right)}\left|i\right\rangle \otimes \left(\sqrt{1-f\left(i\right)}\left|0\right\rangle + \sqrt{f\left(i\right)}\left|1\right\rangle\right). \tag{3.29}$$

3.4 Quantum Machine Learning in Natural Language Processing

NLP has experienced significant progress with the emergence of deep learning techniques capable of extracting complex semantic and contextual information from textual data. However, the increasing complexity of NLP tasks and the high computational cost associated with large-scale models have motivated researchers to investigate alternative paradigms capable of improving learning efficiency and representation capabilities. In this context, QML has emerged as a promising interdisciplinary field combining quantum computing principles with machine learning algorithms to address complex data processing problems [23, 158].

The application of QML to NLP aims to exploit quantum phenomena such as superposition, entanglement, and quantum parallelism in order to enhance feature representation, optimization, and classification performance. Several QML architectures have been proposed and adapted to NLP tasks including sentiment analysis, text categorization, semantic analysis, language modeling, sequence prediction, and information retrieval [137].

Among the earliest QML approaches applied to NLP are Quantum Boltzmann Machines, which extend classical Boltzmann machines using quantum probabilistic mechanisms for modeling hidden semantic distributions [210]. QBMs are particularly suitable for unsupervised learning and latent representation extraction from textual data. Similarly, Variable-Depth Quantum Circuits provide adaptable quantum architectures whose circuit depth can be dynamically adjusted according to task complexity, allowing flexible hybrid quantum-classical learning strategies [80].

Hybrid Quantum Autoencoders have also attracted attention for dimensionality reduction and feature compression in NLP applications. These models enable the extraction of compact semantic representations while preserving essential textual information [172]. In addition, Quantum Reservoir Computing has demonstrated promising capabilities for sequential language processing due to its recurrent dynamic architecture, which is particularly adapted to temporal and contextual textual dependencies [176].

Support Vector Machines with Quantum Kernel Estimators represent one of the most widely explored quantum approaches for NLP classification tasks. By mapping textual features into high-dimensional quantum Hilbert spaces, QSVMs improve feature separability and classification efficiency [74, 156]. Likewise, Quantum Neural Networks and Feedforward Quantum Neural Networks aim to reproduce the behavior of classical neural architectures while leveraging quantum operations for enhanced representation learning [33, 50].

Other quantum-inspired architectures have focused on improving learning robustness and optimization. Orthogonal Neural Networks introduce orthogonality constraints to improve generalization performance and reduce redundancy in learned representations [84, 193]. Quantum Fully Self-Supervised Neural Networks attempt to minimize dependence on labeled datasets by learning intrinsic data structures directly from unlabeled corpora [93]. Similarly, CNOT Neural Networks employ controlled-NOT quantum gates to construct trainable quantum circuits for classification and pattern recognition tasks [103].

Deep-learning-inspired quantum models have become increasingly important in NLP research. Quantum Long Short-Term Memory networks extend recurrent neural architectures to quantum systems and are particularly effective for sequential data processing and contextual dependency learning [31, 34]. Quantum Backpropagation Neural Networks and hybrid quantum feedforward models employ quantum optimization procedures for supervised learning tasks [65, 180].

Generative approaches have also been explored through Quantum Generative Adversarial Networks , which aim to generate synthetic data distributions and improve probabilistic representation learning [6, 167]. Furthermore, Quantum Monte Carlo methods

contribute to optimization and probabilistic inference processes in quantum learning systems [117, 168].

Among the various QML architectures, Quantum Convolutional Neural Networks (QCNNs) have recently emerged as one of the most promising models for NLP classification tasks. Inspired by classical CNNs, QCNNs integrate quantum convolution and pooling operations to extract hierarchical textual features while reducing the number of trainable parameters [43, 81, 203]. Their layered architecture enables efficient processing of encoded textual representations and makes them suitable for hybrid quantum-classical implementations compatible with Noisy Intermediate-Scale Quantum (NISQ) devices [143].

3.4.1 Comparison of Quantum Machine Learning Models applied to NLP

Different QML models exhibit distinct characteristics, advantages, and limitations when applied to NLP tasks. Table 3.1 presents a comparative analysis of the principal QML architectures discussed previously.

Table 3.1: Comparison of QML Models Applied to NLP

Model	**Advantages**	**Limitations**	**Main NLP Applications**
Quantum Boltzmann Machines	Probabilistic semantic modeling	High computational complexity	Language modeling
Variable-Depth Quantum Circuits	Flexible and adaptive architectures	Noise sensitivity	Text classification
Hybrid Quantum Autoencoders	Efficient feature compression	Limited qubit capacity	Representation learning
QSVM with Quantum Kernel	Strong classification performance	Scalability limitations	Sentiment analysis
Quantum Neural Networks	Learns complex feature relationships	Difficult convergence	Text categorization
Orthogonal Neural Networks	Better generalization ability	Expensive optimization	Semantic representation
CNOT Neural Networks	Efficient gate-based operations	Hardware constraints	Pattern classification
QLSTM Networks	Captures contextual dependencies	High circuit complexity	Sequence prediction
Quantum GANs	Synthetic data generation	Difficult training convergence	Text generation
Quantum Monte Carlo	Efficient probabilistic optimization	Computational overhead	Probabilistic NLP models
QCNNs	Hierarchical feature extraction with reduced parameters	Current hardware limitations	Text categorization

From this comparison, it can be observed that QCNNs offer an interesting balance be-

tween classification performance, parameter efficiency, and architectural scalability. Unlike recurrent quantum models that suffer from high circuit complexity, QCNNs provide hierarchical feature extraction while maintaining relatively compact architectures. Moreover, compared to QSVMs and probabilistic quantum models, QCNNs are better adapted to deep feature learning and hybrid implementations on NISQ devices.

3.5 Conclusion

This chapter introduced the main types of QML approaches, including quantum-classical, quantum-quantum, classical-quantum and classical-classical models, demonstrating how quantum computation can be integrated into learning processes to improve computational efficiency and data representation.

The chapter also explored several important QML models and architectures, such as quantum neural networks, quantum support vector machines, variational quantum circuits, and quantum kernel-based methods. These models illustrate the growing potential of quantum systems in solving complex optimization and pattern recognition problems while addressing some limitations of classical machine learning approaches.

Despite the promising capabilities of QML, the field remains in an early stage of development due to current hardware limitations and challenges related to scalability, noise, and quantum resource optimization. Nevertheless, ongoing advances in quantum technologies and hybrid learning frameworks continue to expand the applicability of QML across different domains, including NLP.

Building upon the theoretical foundations presented in this chapter, the next chapter focuses on the practical application of QCNN in NLP, with particular emphasis on sentiment analysis tasks. The integration of QCNN architectures with NLP techniques represents a promising direction for exploring the potential advantages of quantum-enhanced models in text classification and language understanding applications.

Chapter 4

Application of QCNN to Sentiment Analysis

4.1 Introduction

In this chapter, we list the contributions of our work as follows:

- Developing a QCNN-based method for sentiment analysis that combines the strengths of quantum machine learning and classical convolutional neural networks.
- Introducing three QCNN models that differ in their number of qubits, implemented using both the Pennylane and Qiskit frameworks and built with predefined embedding and ansatz techniques.
- Evaluating the proposed models using precision, accuracy and F1 score, and conducting comparisons to highlight their differences.

4.2 Related work

QML holds significant potential across a wide range of applications, including sentiment analysis. Table 4.1 summarizes how QML has been applied to sentiment analysis in various studies.

Table 4.1: The table highlights representative studies in sentiment analysis employing quantum-inspired machine learning or QML.

Approach	Quantum encoding	Dataset	Author
Quantum-inspired Interactive Networks(QIN)	not available	IEMOCAP, MELD	Zhang et al. [206]
GQLM	not available	SS-tweet, OMD	Zhang et al. [205]
TextTN	not available	CR, MR, MPQA, Subj	Zhang et al. [207]
QNet	Positional encoding	ColBERT	Day et al. [45]
TextTN + BERT	not available	SST-5, SST-2	Zhang et al. [207]
QFNN	Angle encoding	CVTD , GSTD	Dave et al.[44]

As previously noted, QCNNs constitute one of the central architectures within Quantum Machine Learning. Although their use has been investigated in several areas—such as image classification—their application to sentiment analysis remains unexplored. The QCNN framework was originally introduced by [43], who adapted the classical CNN architecture to the quantum computing paradigm. In the present work, the QCNN model executes a sequence of convolutional and pooling unitaries, which form the core building blocks of the architecture. We summarize the overall procedure as follows:

1. The convolution layer derives the hidden state through the application of multi-qubit gates across adjacent qubits.

2. The pooling circuit reduces the size of the quantum system by applying two-qubit gates.

3. Iterate the convolution and pooling unitaries described in steps (1)–(2).

4. After the system has been downscaled through previous layers, the fully connected layer produces the final classification output.

The architecture is based on the Multiscale Entanglement Renormalization Ansatz (MERA) [188]. In the context of QCNNs, MERA is implemented in reverse, allowing the quantum system to be reduced exponentially in size as a function of the input data. Since the introduction of QCNNs, this framework has been widely adopted across various applications. In particular, [81] proposed a QCNN architecture for image classification that relies exclusively on two-qubit interactions, achieving high classification accuracy with a notably small set of trainable parameters. Their study systematically evaluates the performance of parameterized QCNNs under different conditions using the Fashion-MNIST and MNIST datasets on PennyLane. The study also investigates multiple quantum data encoding strategies including dense, qubit, angle, amplitude and hybrid encoding while accounting for dimensionality reduction techniques, which are well suited to early quantum devices with limited qubit counts. All QCNN configurations demonstrated strong classification performance, reaching accuracies of 99% on MNIST and about 94% on Fashion-MNIST. Furthermore, comparisons with classical CNNs showed that QCNNs not only delivered competitive results but, in some cases, surpassed the performance of their classical counterparts.

In the present work, we introduce a new QCNN-based framework for sentiment analysis. Word representations are generated using Word2Vec, and three quantum encoding methods which are angle encoding, amplitude encoding and IQP encoding. In addition, some of the convolution and pooling unitaries from the QCNN model in [81] are adapted to suit our architecture.

4.3 Methods

We employ QCNNs on a movie review dataset to classify sentences as either positive or negative. The general architecture of this QCNN-based classifier is presented in Figure 4.1.

The following subsections detail each component of the architecture and describe how they collectively enable the QCNN's functionality.

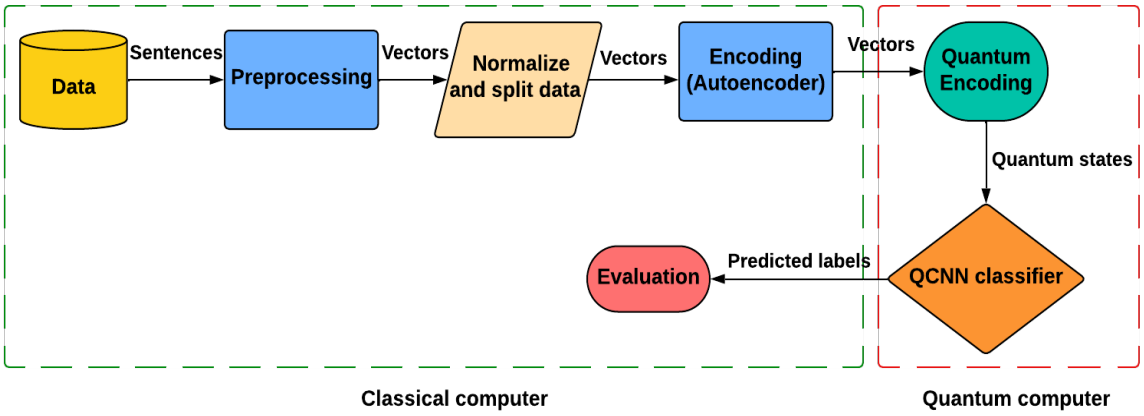


Figure 4.1: General structural layout of the model.

4.3.1 Preprocessing

Once the data is collected, a two-stage process is applied, comprising preprocessing and vector generation. The preprocessing step converts the raw textual input into a cleaner and more structured form suitable for analysis. Sentiment analysis datasets are often large and may include irregular entries due to the diversity of their sources. Consequently, the preprocessing stage involves tokenizing the text, removing punctuation, stopwords, and single-character tokens, and converting all text to lowercase. After this cleaning process, the text is prepared for vector representation. The second stage, known as vector generation or word embedding, maps textual tokens to numerical vectors that can be effectively processed by machine learning models. This work uses Word2Vec embeddings, representing each word as a 300-dimensional vector learned through a neural network–based model. It supports two efficient architectures, Skip-gram and Continuous Bag of Words (CBOW) both implemented using a shallow neural network with a single hidden layer. The process starts with the construction of a vocabulary from the training text, and the word vectors are learned via backpropagation and stochastic gradient descent [163].

4.3.2 Standardize and partition the data

Dataset standardization, often referred to as centering the data, consists of transforming the feature values such that the resulting distribution has a mean of 0 and a standard

deviation of 1. This is accomplished by subtracting the mean of each feature from every observation and then dividing by the corresponding standard deviation. Standardization is especially important when the input features operate on different scales, and it is typically applied under the assumption that the data exhibit a Gaussian distribution with a clearly defined mean and variance. After standardization, the dataset is divided into two subsets: a training set and a testing set. In our case, the training set includes 67% of the total reviews, while the remaining 33% is reserved for testing.

4.3.3 Dimensionality reduction

The reliable execution of large-scale quantum circuits on Noisy Intermediate-Scale Quantum (NISQ) devices is limited by hardware noise and other technological constraints. Consequently, the encoding of high-dimensional datasets often requires a number of qubits that exceeds the capacity of current quantum processors. To mitigate this issue, classical dimensionality reduction techniques can be applied to compress the data before quantum encoding, thereby enabling the practical deployment of QML algorithms on near-term quantum hardware.

In the present work, dimensionality reduction of the word vectors is a crucial preprocessing step to ensure their compatibility with the QCNN-based classification model. By reducing the feature space, the resulting representations can be efficiently encoded into the available quantum resources while preserving the most relevant information for the classification task.

Among the various dimensionality reduction techniques, autoencoders offer an efficient and widely used solution. An autoencoder comprises two principal components: an encoder and a decoder. The encoder maps the input data into a lower-dimensional latent space, while the decoder attempts to reconstruct the original input from this compressed representation as faithfully as possible. By learning to extract the most relevant features from the data, the encoder effectively reduces dimensionality while filtering out noise and redundancy. In our approach, the initial 300-dimensional word embeddings are first compressed to match the input dimensionalities required by the QCNN classifier (4, 8, or 16). This dimensionality reduction is achieved using an autoencoder-based architecture

trained in an unsupervised manner to reconstruct the original input data. The model employs the Adam optimizer and the Mean Squared Error (MSE) to learn compact latent representations that preserve the most relevant information in the embedding space. The autoencoder consists of a single hidden layer, where the latent space dimension is set according to the desired compression level (4, 8, or 16). After training, the encoder component is retained to generate the reduced feature vectors, as depicted in Figure 4.2. This step ensures compatibility with the limited number of qubits available in the QCNN while preserving the essential semantic structure of the original word embeddings.

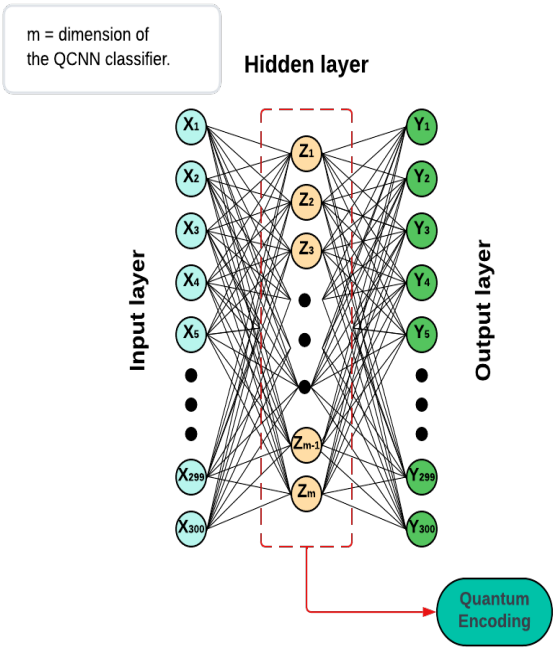


Figure 4.2: Architecture of the model's autoencoder.

The performance of the autoencoder is evaluated across different latent dimensionalities to analyze the trade-off between compression and reconstruction quality. For a latent dimension of 4, the model achieves a final training loss of approximately 1.18 and a validation loss of approximately 1.22, indicating the strongest compression but also the highest reconstruction error among the tested configurations. Increasing the effective encoding capacity to the latent dimension of 8 improves performance, yielding a validation loss of approximately 1.20 and demonstrating more stable generalization behavior. Further increasing the latent dimension to 16 results in the best reconstruction performance,

with a validation loss of approximately 1.15. These results highlight a clear trade-off between dimensionality reduction and reconstruction fidelity, where lower-dimensional representations reduce quantum resource requirements but incur higher information loss, whereas higher-dimensional latent spaces preserve more semantic information at the cost of increased encoding complexity.

The autoencoder generates latent representations in an unconstrained real-valued space $\mathbb{R}^d$, where $d \in \{4, 8, 16\}$ denotes the dimensionality of the latent space. These representations are not restricted to any predefined numerical interval and may take both positive and negative values depending on the learned embedding distribution. Similarly, the decoder reconstructs the input embeddings in $\mathbb{R}^{300}$ without explicit bounding constraints, as the model is optimized solely through a reconstruction loss function.

The min-max scaling step is specifically required for angle encoding, where classical features are mapped to rotation angles of quantum gates. This transformation ensures that all input values lie within a bounded interval such as $[0, \pi]$, which is necessary for stable quantum circuit execution. In contrast, this scaling is not required for alternative encoding strategies such as amplitude encoding, which relies on ℓ_2 normalization of feature vectors, or IQP-based encodings, which typically operate on phase values.

4.3.4 Quantum encoding

Following dimensionality reduction, the vectors are transformed into quantum states via an appropriate quantum encoding method. Below, we present different quantum encoding methods used in this study with a comprehensive description of each approach .

Angle encoding

Owing to its simplicity, this approach is one of the most frequently used. It exploits the dependence of measurement probabilities on the state's phase. Quantum states are expressed as complex numbers, with amplitudes encoding measurement probabilities and phase angles playing a key role in quantum algorithms. Angle encoding allows classical data to be mapped into quantum states, enabling quantum computers to efficiently process this information. For example, the angle embedding operation of an angle θ and a qubit

state $|\psi\rangle$ is given by:

$$\left|\vec{x}^{(j)}\right\rangle = \bigotimes_{k=1}^{N} \left(\cos\left(\vec{x}_k^{(j)}\right)|0\rangle + \sin\left(\vec{x}_k^{(j)}\right)|1\rangle\right). \tag{4.1}$$

This method is well-suited for sentiment analysis, as it retains the vector space relationships essential for detecting context-dependent sentiment.

Amplitude encoding

In this method, normalized input data are mapped onto the probability amplitudes of a quantum state. Given an input vector

$$x = (x_1, x_2, \ldots, x_N)^T,$$

where $N = 2^n$, it is encoded as the amplitudes of an n-qubit quantum state $|\phi(x)\rangle$.

$$U_\phi(x) : x \in \mathbb{R} \rightarrow |\phi(x)\rangle = \frac{1}{||x||}\sum_{i=1}^{N} x_i |i\rangle.$$

Here, $|i\rangle$ denotes the ith computational basis state. This encoding method efficiently represents large classical datasets on a quantum computer, providing a key advantage for QCNNs. As the number of parameters scales with $O(\log(n))$, this approach drastically reduces the parameter count relative to the data dimension.

Amplitude encoding is adopted due to its widespread use and effectiveness in embedding dense numerical features, like word vector magnitudes or sentiment polarity scores. However, the compression inherent in amplitude squashing may obscure subtle emotional nuances in the data.

IQP encoding

The IQP encoding, based on physical principles, maps data $x \in (0, 2\pi]^n$ into a second-order Hamiltonian and can offer a quantum advantage. The function f is defined as: $\tilde{f} = 2\pi\mathring{u}f$ that maps $\tilde{f} : \mathbb{R}^{n-1} \rightarrow (0, 2\pi]^n$ preserves information, as it is a homeomorphism. Denoting Z_i as the Pauli-Z gate acting on the ith qubit, the authors define the corresponding unitary operator as:

$$U_Z(x) = \exp(\sum_{i=1}^{n} x_i Z_i + \sum_{i=1}^{n}\sum_{j=1}^{n}(\pi - x_i)(\pi - x_j)Z_i Z_j).$$

It can be seen that the coefficients of the quadratic terms are distributed around 0 with a standard deviation of 1, enabling the original data to be mapped into a transformed representation. Letting H denote the Hadamard gate, the IQP encoding is then defined as:

$$D_i \to U_Z(D_i)H^{\oplus n}U_Z(D_i)H^{\oplus n}\left|0...0\right\rangle .$$

IQP encoding is especially useful for representing non-linear and entangled relationships within textual data, enabling a more expressive encoding of complex sentiment patterns, including subtle phenomena like irony and sarcasm.

Comparative Analysis

In this subsection, we present a comparative analysis of the quantum encoding methods discussed above. (refer to [158, 30, 23, 178, 156, 74, 143]).

As shown in Table 4.2, while amplitude encoding is efficient and straightforward, it may struggle to represent the fine-grained emotional nuances present in text. Angle encoding offers a balanced approach, maintaining interpretability while better preserving sentiment, particularly when working with vectorized representations such as word embeddings. IQP encoding, while more challenging to implement, offers significant potential for capturing complex linguistic structures and nuanced sentiment variations such as sarcasm or subtle emotional cues, as it can represent them through highly entangled and complex quantum states.

4.3.5 QCNN classification

Ansatz

An essential step in the design of a Quantum Convolutional Neural Network (QCNN) model is the selection of an appropriate *ansatz*. In quantum machine learning, an ansatz refers to a predefined parameterized quantum circuit architecture that specifies the arrangement of quantum gates, entanglement patterns, and trainable parameters used to represent the model. Similar to the architecture of a classical neural network, the ansatz

Table 4.2: Comparison of quantum data encoding techniques

Criterion	Angle Encoding	Amplitude Encoding	IQP Encoding
Implementation Complexity	Moderate	Low to Moderate	High
Encoding Efficiency	Moderate	High (needs normalization)	Low
Sentiment Feature Retention	High (effectively captures relative context)	Low to Moderate	High
Information Loss	Low	Moderate (due to amplitude compression)	Low to Moderate
Suitability for Emotionally-Rich Text	Good	Limited	Very Good

determines the representational capacity of the QCNN and influences its ability to learn meaningful patterns from data.

In principle, the QCNN architecture is sufficiently flexible to employ arbitrary two-qubit unitary operations at each convolutional filter and pooling stage. However, in this work, the model design is constrained such that all convolutional filters share an identical ansatz, while all pooling operations similarly adopt a common ansatz distinct from that of the convolutional layers.

This design choice is motivated by the need to control the model complexity and reduce the number of trainable parameters. Although assigning distinct ansatz structures to individual filters may offer potential improvements in representational capacity and predictive performance, such an approach would significantly increase the optimization complexity due to the larger parameter space.

QCNN architecture

This section presents three QCNN classifiers, differing in input dimensions and layer counts, as illustrated in Figures 4.3, 4.4, and 4.5. Each QCNN classifier starts with word embeddings that are encoded into quantum states. The quantum states then undergo repeated convolution and pooling operations until the system is reduced to a single qubit. Finally, measurements are performed on all qubits to convert the quantum states back into classical data for subsequent processing.

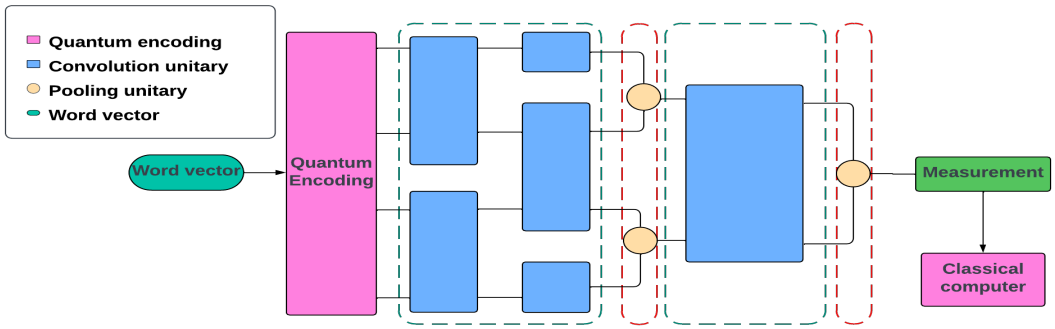


Figure 4.3: QCNN architecture with a 4-qubit input.

The quantum circuit illustrated in Figure 4.3 comprises three primary components: quantum data encoding (pink rectangles), convolutional unitaries (blue rounded rectangles), and pooling unitaries (orange circles). Parameterized quantum gates are employed to implement convolutional and pooling operations. The QCNN comprises two layers, each hosting several convolutional unitaries that apply a shared two-qubit ansatz to nearest-neighbor qubits in a translationally invariant way. Pooling operations use the same ansatz as controlled unitaries, triggered when the control qubit is in state $|1\rangle$.

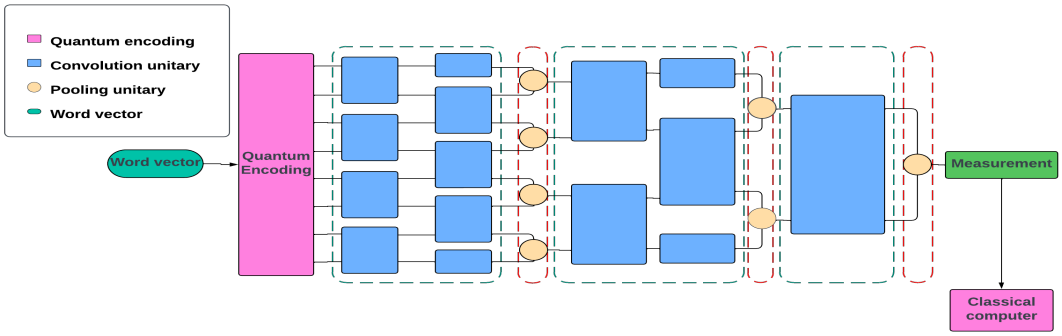


Figure 4.4: QCNN architecture with a 8-qubit input.

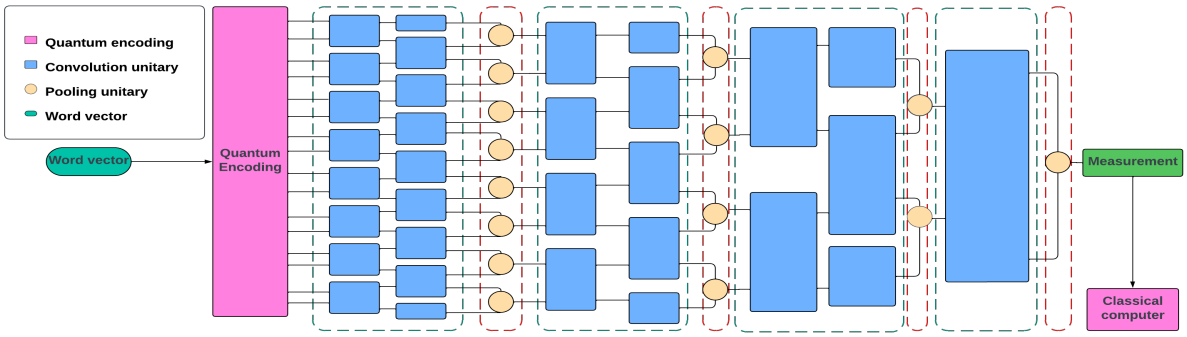


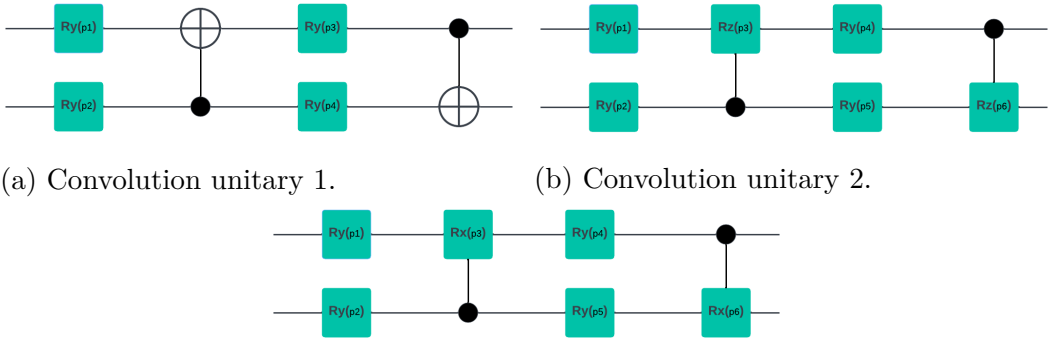
Figure 4.5: QCNN architecture with 16-qubit input.

The QCNN architecture comprises two primary components:

1. Convolution layer. Each QCNN employs a two-qubit unitary ansatz. While different ansätze could be used per convolutional filter, here we apply the same ansatz uniformly across all convolutional layers, sacrificing possible performance improvements for simplicity, it would also expand the set of parameters to be optimized. We employ three different unitaries for convolution, each chosen to entangling capability and balance expressibility, as discussed in [81]. The architecture of each unitary is illustrated in Figure 4.6.

2. Pooling layer. This component performs a parameterized pooling unitary on two qubits, followed by the removal of one qubit, resulting in a single-qubit state. Although multiple designs for the pooling unitary exist, in this work we adopt a simplified version following the approach of [81]. The architecture of this unitary is depicted in Figure 4.7.

This component performs two controlled rotations, $R_z(p_1)$ and $R_z(p_2)$, which operate only when the control qubit is prepared in the state 1 (filled circle) or 0 (open circle),



(a) Convolution unitary 1.

(b) Convolution unitary 2.

(c) Convolution unitary 3.

Figure 4.6: Convolution unitaries.

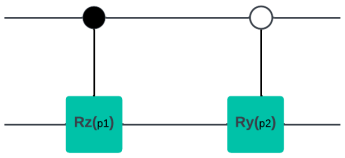


Figure 4.7: Pooling unitary.

respectively. After these gate operations, the first qubit is traced out to achieve dimensionality reduction.

4.3.6 Cross-Entropy Loss

This loss function is widely employed in the training of classical neural networks as an objective function for classification tasks. It evaluates the performance of a classification model by measuring the discrepancy between the predicted probability distribution and the true class labels, where the model output is represented as a probability value between 0 and 1. Owing to the probabilistic nature of quantum mechanics, the cross-entropy loss can also be adapted to quantum neural network architectures, such as the Quantum Convolutional Neural Network (QCNN), by considering the probabilities associated with measurements in the computational basis.

In the context of QCNNs, the cross-entropy loss is defined using the probabilities obtained from single-qubit measurements of computational basis states. For the i-th training sample, the loss function can be expressed as

$$C(\theta) = -\sum_{i=1}^{N} \left[y_i \log \left(\Pr \left[\psi_i(\theta) = 1 \right] \right) + (1 - y_i) \log \left(\Pr \left[\psi_i(\theta) = 0 \right] \right) \right], \tag{4.2}$$

where $y_i \in \{0, 1\}$ denotes the true class label, and $\Pr[\psi_i(\theta) = y_i]$ represents the probability of measuring the computational basis state corresponding to the correct class label from the QCNN circuit parameterized by θ. This formulation enables the optimization process to adjust the trainable quantum circuit parameters in a manner that maximizes the likelihood of correct classification.

4.3.7 Adam Optimizer

Adaptive Moment Estimation is a stochastic optimization algorithm widely used for training deep learning models. It combines the advantages of Momentum and RMSProp by adaptively adjusting the learning rate for each model parameter using estimates of the first and second moments of the gradients.

The parameter update rule of Adam is defined as:

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon},$$

where η represents the learning rate, $\hat{m}_t$ and $\hat{v}_t$ denote the bias-corrected first and second moment estimates, respectively, and ϵ is a small constant added for numerical stability.

Adam was selected in this work because of its fast convergence, computational efficiency, and robustness when handling sparse and noisy gradients. Furthermore, it requires limited memory and minimal hyperparameter tuning, making it suitable for NLP.

4.4 Results

This section presents the results of the classical simulation of the QCNN algorithm for binary classification, conducted using different computational resources, which are described in the following section. The experiments focus on short-sentences of movie reviews, where binary classification is adopted as the benchmark task. Specifically, the objective is to classify movie reviews into two sentiment categories (negative or positive).

For all experiments, approximately two-thirds of each dataset was used for training, while the remaining one-third was reserved for testing. Prior to training, all input sentences were preprocessed to ensure consistency and suitability for quantum encoding. In addition, the publicly available pre-trained Google News *Word2Vec* model was employed to generate vector representations of words, enabling textual data to be transformed into numerical feature embeddings compatible with the QCNN framework.

The evaluation was conducted on two benchmark datasets, referred to as the *artificial* and *real* datasets, both formulated as binary classification problems and they contain two classes featuring an equitable distribution of positive and negative reviews

1. Artificial dataset. The dataset employed in this study was introduced in the work of [17]. The dataset consists of short sentences containing a maximum of seven words. A total of 134 sentences were used for training, while 66 sentences were reserved for testing.

2. Real dataset. The dataset corresponds to the *Sentence Polarity Movie Review* dataset obtained from *imdb.com*. This dataset contains movie review sentences with a maximum sentence length of 59 words. For the purposes of this study, $7,137$ sentences were used for training and $3,516$ sentences were allocated for testing.

The variational parameters of the QCNN ansatz were optimized by minimizing the cross-entropy cost function using optimization tools provided by the *Pennylane* framework. In particular, the *Adam* optimizer was employed to update the trainable parameters. At each training iteration t, a mini-batch of samples was randomly selected from the training dataset.

Compared with full-batch training, the mini-batch strategy reduces computational cost and simulation time while improving optimization dynamics by helping gradients avoid poor local minima. In this study, the batch size was fixed at 25, and the learning rate was set to 0.01. Furthermore, the number of training iterations was limited to 200 in order to reduce computational overhead and accelerate the training process.

4.4.1 Programming resources

The implementation and execution of quantum machine learning models are computationally demanding and often require significant processing time. To conduct the experi-

ments presented in this work, several programming tools and computational frameworks were employed, including *Jupyter Notebook*, *Pandas*, *Qiskit*, *TensorFlow*, and *Pennylane*. These tools were utilized for data preprocessing, model implementation, quantum circuit simulation, optimization, and result analysis.

Jupyter Notebook

It is an interactive, notebook-based development environment designed for writing, executing, and analyzing code efficiently. It is widely used in data science, scientific research, and educational applications due to its flexibility in integrating code execution, data visualization, and documentation within a unified workspace.

Pandas

It is a powerful Python library used for data manipulation, organization, and analysis. It provides efficient data structures and tools for cleaning, transforming, and processing structured data, making it particularly suitable for data preprocessing and exploratory data analysis in scientific and machine learning applications.

Qiskit

It is an open-source Python-based software development toolkit developed by IBM for designing, simulating, and executing quantum computing programs. It enables the implementation of quantum circuits on both simulated environments and real quantum hardware, thereby providing a comprehensive framework for the development and evaluation of quantum algorithms.

Tensorflow

It is an open-source machine learning framework widely used for the development, training, and deployment of machine learning and deep learning models. It provides a flexible computational architecture for building neural networks, optimizing model performance, and handling large-scale numerical computations efficiently.

Pennylane

PennyLane is a software framework that bridges quantum computing and machine learning, allowing users to design, train, and run quantum experiments on simulators as well as real quantum hardware.

4.4.2 Experimental configuration

The QCNN model was implemented using TensorFlow, PennyLane, and Qiskit. The hardware specifications employed for the experiments are summarized in Table 4.3.

Table 4.3: Execution environment hardware. The label n/a refers to entries where information is not available.

Item	Detail
OS	Microsoft Windows 11, 64 bit
RAM	16 GO
CPU	AMD Ryzen 74700U with Radeon Graphics
GPU	n/a
TPU	n/a

4.4.3 Evaluation of QCNN performance using the artificial dataset

Table 4.4 reports the highest classification precision, accuracy and F1 score obtained on the artificial dataset across the different QCNN classifiers using random parameter initialization. The highest value in each column is highlighted in bold.

Table 4.4: Classification performances achieved by QCNN on the artificial dataset.

Number of qubits	Ansatz	Number of params	Quantum encoding	Accuracy	Precision	F1 score	Runtime (Seconds)
4	1	12	Angle	83,3%	84,6%	83,3%	0,95
	2	16	Angle	89,4%	89,8%	89,3%	1,00
	3	16	Angle	**90,9%**	**90,9%**	**90,9%**	1,29
	1	12	Amplitude	63,6%	63,7%	62,8%	0,89
	2	16	Amplitude	63,6%	63,5%	63,3%	0,97
	3	16	Amplitude	**69,7%**	**69,6%**	**69,4%**	1,23
	1	12	IQP	78,8%	82,7%	78,5%	1,00
	2	16	IQP	72,7%	77,8%	70,5%	1,07
	3	16	IQP	**83,3%**	**85,1%**	**82,9%**	1,26
8	1	18	Angle	92,4%	92,5%	92,4%	2,44
	2	24	Angle	81,8%	84,6%	81,7%	2,53
	3	24	Angle	**92,4%**	**93,0%**	**92,3%**	2,88
	1	18	Amplitude	63,6%	63,5%	63,5%	2,10
	2	24	Amplitude	65,2%	65,3%	65,1%	2,30
	3	24	Amplitude	**66,7%**	**66,7%**	**66,6%**	2,83
	1	18	IQP	81,8%	82,7%	81,8%	2,72
	2	24	IQP	84,8%	87,8%	84,7%	2,74
	3	24	IQP	**86,4%**	**88,7%**	**86,3%**	3,69
16	1	24	Angle	87,9%	88,8%	87,9%	19,97
	2	32	Angle	**90,9%**	**90,9%**	**90,9%**	23,66
	3	32	Angle	89,4%	89,4%	89,3%	26,44
	1	24	Amplitude	/	/	/	/
	2	32	Amplitude	/	/	/	/
	3	32	Amplitude	/	/	/	/
	1	24	IQP	71,2%	75,0%	70,7%	33,29
	2	32	IQP	80,3%	80,5%	80,1%	35,36
	3	32	IQP	**90,9%**	**91,0%**	**90,8%**	37,22

4.4.4 Evaluation of QCNN performance using the real dataset

Table 4.5: Classification performances achieved by QCNN on real dataset.

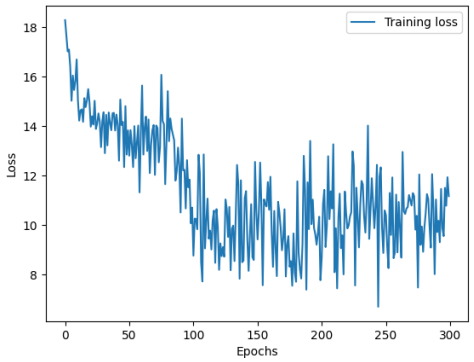
Number of qubits	Ansatz	Number of params	Quantum encoding	Accuracy	Precision	F1 score	Runtime (Seconds)
4	1	12	Angle	**52, 0%**	**52, 0%**	**52, 0%**	478, 36
	2	16	Angle	51, 8%	51, 8%	51, 7%	555, 13
	3	16	Angle	51, 8%	51, 8%	51, 8%	667, 05
	1	12	Amplitude	51, 2%	51, 2%	51, 2%	481, 84
	2	16	Amplitude	**51, 3%**	**51, 3%**	**51, 3%**	537, 72
	3	16	Amplitude	50, 6%	50, 6%	50, 6%	600, 99
	1	12	IQP	51, 4%	51, 5%	50, 0%	544, 71
	2	16	IQP	**51,5%**	**51,5%**	**51,1%**	1329, 87
	3	16	IQP	51, 1%	51, 5%	48, 2%	1653, 99
8	1	18	Angle	**53,3%**	**53,5%**	**51,2%**	737, 31
	2	24	Angle	53, 2%	53, 4%	50, 2%	3484, 80
	3	24	Angle	53, 1%	53, 0%	52, 1%	3318, 79
	1	18	Amplitude	**51, 7%**	**51, 7%**	**51, 6%**	681, 24
	2	24	Amplitude	51, 3%	51, 3%	51, 1%	1346, 31
	3	24	Amplitude	51, 1%	51, 1%	51, 1%	2104, 01
	1	18	IQP	**51,7%**	**51,7%**	**51,5%**	972, 95
	2	24	IQP	50, 8%	51, 4%	46, 3%	3578, 74
	3	24	IQP	51, 0%	51, 0%	50, 7%	3867, 56
16	1	24	Angle	51, 8%	51, 8%	50, 9%	11469, 96
	2	32	Angle	**51,8%**	**51,8%**	**51,8%**	12050, 21
	3	32	Angle	51, 0%	43, 2%	50, 4%	12150, 64
	1	24	Amplitude	/	/	/	/
	2	32	Amplitude	/	/	/	/
	3	32	Amplitude	/	/	/	/
	1	24	IQP	51, 2%	51, 2%	50, 7%	12030, 96
	2	32	IQP	50, 6%	51, 5%	43, 4%	13430, 96
	3	32	IQP	**51,9%**	**52,0%**	**51,7%**	13887, 96

The highest classification precision, accuracy, and F1 score obtained on the artificial dataset for each QCNN classifier with random parameter initialization are summarized in Table 4.4. The best performance in each column is highlighted in bold.

4.4.5 Evolution of QCNN classification

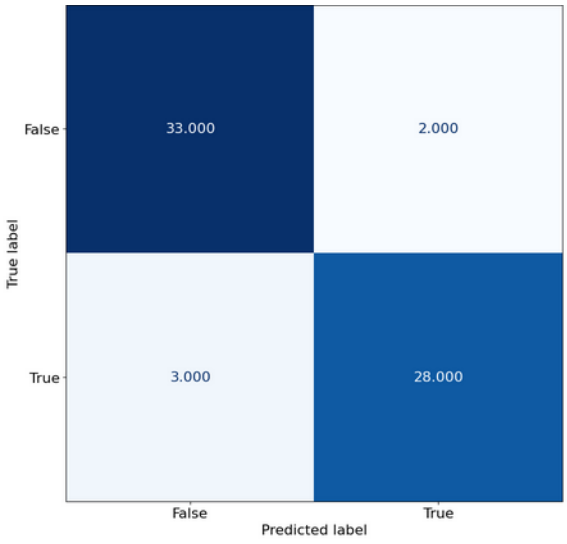
We run the QCNN of 8 qubits in artificial dataset multiple time and we plot the best evolution of the cost function on the training dataset in Figure 4.8.

Figure 4.8: Evolution of the cross entropy on training set with number of epochs 300 and batch size=25.



We plot the results in the form of normalized confusion matrix, see Figure 4.9 .

Figure 4.9: confusion matrix



We see that the training cost function exhibited an overall decreasing trend during the initial epochs, indicating successful learning of sentiment-related representations by the QCNN model. A rapid reduction in loss was observed during the early training phase, followed by a region of lower loss values around epochs 100–180, suggesting effective optimization. However, fluctuations in later epochs indicate optimization instability, which is common in variational quantum neural networks due to complex parameter landscapes and gradient sensitivity. Despite this instability, the confusion matrix demonstrated strong classification capability, achieving approximately 92.4% accuracy with only five misclassifications. This suggests that the QCNN successfully learned robust sentiment decision boundaries even without smooth loss convergence.

4.4.6 Interpretation of experimental findings

Training a 4-qubit QCNN on the artificial dataset shows that Ansatz 3 achieves the highest accuracy of 90.9% when using angle encoding. Expanding to an 8-qubit QCNN improves classifier performance relative to the 4-qubit model. Among the tested ansatz, Ansatz 3 again delivers the best accuracy of 92.4% with angle encoding, indicating that angle encoding consistently outperforms other quantum encoding methods for both 4- and 8-qubit configurations, while amplitude encoding produces lower and less stable accuracies. As summarized in Table 4.4, the 16-qubit QCNN achieves the best overall performance among all tested models, with angle encoding once more proving to be the most effective approach. This demonstrates that increasing the number of qubits generally enhances accuracy, and that Ansatz 3 consistently provides superior results. It should be noted that amplitude encoding cannot be applied to the 16-qubit QCNN, as $2^{16} > 300$, exceeding the dimensionality of the word vectors. For the real dataset, QCNN accuracies are generally similar across different encoding methods and ansatz. However, for 4- and 8-qubit QCNNs, Ansatz 1 combined with angle encoding achieves the highest accuracies, 52.0% and 53.3%, respectively. In the 16-qubit QCNN, Ansatz 3, when paired with IQP encoding, reaches a maximum accuracy of 51.9%. Overall, angle encoding still demonstrates better performance compared to other quantum encoding strategies. Table 4.5 shows that the results are converging, suggesting that variations in encoding methods,

QCNN architecture, or ansatz have only a limited impact on performance. Comparing Tables 4.4 and 4.5, The artificial dataset yields higher accuracies than the real dataset, which is unsurprising given that its sentences are much shorter (7 tokens) compared to the longest sentences in the real dataset (59 tokens). Shorter sequences generally allow classifiers to perform more effectively. Moreover, the number of qubits does not appear to significantly influence accuracy, consistent with the QCNN's ability to generalize even with relatively shallow architectures. Finally, Tables 4.4 and 4.5 also highlight that runtime increases with the number of qubits due to larger circuit sizes and more parameters. Among encoding methods, IQP encoding is slower, whereas amplitude encoding is faster but less effective in terms of accuracy.

4.5 Conclusion

This chapter presented the design, implementation, and evaluation of a QCNN applied to sentiment analysis of movie reviews, with a focus on assessing the impact of quantum architecture design choices and encoding strategies on classification performance. By systematically varying the number of qubits, circuit ansatz, and encoding methods, the study provides a detailed empirical investigation into how these factors influence QCNN behavior across both artificial and real-world datasets.

The experimental results on the artificial dataset demonstrate that QCNNs can achieve high classification performance, with accuracy improving as the number of qubits increases. In particular, Ansatz 3 combined with angle encoding consistently yields the best results, reaching up to 92.4% accuracy in the 8-qubit configuration and remaining the most effective architecture across different scales. Angle encoding also proves to be the most stable and effective representation method, while amplitude encoding shows lower performance and limited applicability due to dimensionality constraints.

In contrast, experiments conducted on the real dataset reveal significantly lower accuracies and reduced sensitivity to architectural variations. Although angle encoding remains slightly more effective overall, performance differences between ansatz and encoding strategies are minimal, indicating a convergence in model behavior. This suggests that real-world textual data introduces additional complexity that limits the discrimina-

tive advantage of current QCNN architectures.

The comparison between datasets highlights a clear dependence of QCNN performance on input structure and complexity. The higher accuracy achieved on the artificial dataset can be attributed to shorter sentence lengths and reduced linguistic variability, whereas the longer and more diverse sentences in the real dataset pose greater challenges for quantum models. Additionally, the study shows that increasing the number of qubits does not guarantee improved performance in real-world scenarios, reinforcing the idea that QCNN effectiveness is highly data-dependent.

Finally, the analysis of computational cost confirms that larger quantum circuits lead to increased runtime due to higher circuit complexity and parameter counts. Among encoding techniques, IQP encoding is the most computationally expensive, while amplitude encoding is faster but less effective in practice. In addition, the effectiveness of QCNN remains sensitive to dataset characteristics and encoding design, highlighting the need for further research to improve scalability, robustness, and applicability to complex real-world NLP tasks.

Conclusion and perspectives

This thesis has explored the emerging intersection between Natural Language Processing (NLP), quantum computing, and Quantum Machine Learning (QML), with the objective of investigating the potential of quantum-enhanced models for text classification tasks, particularly sentiment analysis. The work is structured progressively, beginning with the foundational principles of NLP, moving through quantum computing and QML theories, and culminating in a practical implementation of a Quantum Convolutional Neural Network (QCNN) for sentiment classification of movie reviews.

The first chapter established the theoretical and conceptual foundations of NLP. It presented the key components of Natural Language Understanding (NLU) and Natural Language Generation (NLG), and traced the evolution of the field from rule-based and symbolic systems to statistical and deep learning approaches. The chapter highlighted major milestones such as word embeddings, recurrent architectures, convolutional models, and transformer-based systems like BERT, which have significantly advanced contextual language understanding. It also reviewed major NLP applications, including sentiment analysis, machine translation, and information extraction. Despite these advances, the chapter emphasized persistent challenges such as computational complexity, semantic ambiguity, and scalability, which motivate the exploration of alternative computational paradigms.

The second chapter introduced the fundamental principles of quantum computing, providing the theoretical background necessary for understanding quantum-enhanced computation. Core concepts such as qubits, superposition, entanglement, quantum interference, and quantum circuits were presented as the basis of quantum information processing. The chapter also examined key quantum algorithms, including Shor's and Grover's algorithms, as well as hybrid approaches such as the Variational Quantum Eigensolver (VQE). These

developments demonstrated the potential of quantum computing to outperform classical approaches in specific computational tasks, particularly in optimization and search problems, thereby laying the groundwork for quantum machine learning applications.

Building upon this foundation, the third chapter explored the field of Quantum Machine Learning, which integrates quantum computing with classical learning frameworks. Different QML paradigms were presented, including quantum-classical and hybrid architectures, alongside major models such as quantum neural networks, quantum support vector machines, variational quantum circuits, and quantum kernel methods. While these approaches show significant promise in enhancing computational efficiency and representation learning, the chapter also highlighted important limitations related to hardware constraints, noise, and scalability, particularly in the current NISQ (Noisy Intermediate-Scale Quantum) era.

The final contribution chapter focused on the practical application of QCNNs to sentiment analysis of movie reviews. This work represents a systematic exploration of quantum convolutional architectures in an NLP context, with particular attention to the effects of qubit configuration and encoding strategies. Experimental results demonstrated that QCNNs are capable of performing sentiment classification tasks effectively, showing strong generalization ability even with shallow quantum circuits. However, performance variations between artificial and real-world datasets highlighted the sensitivity of quantum models to data characteristics such as sentence structure and length. Despite their computational cost and current scalability limitations, QCNNs were shown to be a viable and promising approach for quantum-enhanced NLP tasks.

Overall, this thesis demonstrates that the integration of quantum computing with NLP through QML techniques represents a promising but still developing research direction. While current limitations in hardware and optimization restrict large-scale deployment, the results obtained in this study confirm the feasibility and potential advantages of quantum-enhanced models in text classification tasks. Future research may focus on improving quantum encoding strategies, optimizing variational circuits, exploring larger and more diverse datasets, and extending QCNN architectures to multi-class and more complex NLP problems. In addition, this work can be extended by investigating other

word embedding techniques in order to further improve textual representation and model performance. Although Word2Vec provided satisfactory results in terms of semantic representation and computational efficiency, more advanced embedding approaches such as ELMo, and BERT could be explored.

In conclusion, this work contributes to the growing field of Quantum Natural Language Processing by providing both a theoretical foundation and an initial empirical investigation into the application of QCNNs for sentiment analysis. It highlights the potential of quantum machine learning as a transformative paradigm for future intelligent language-processing systems, while also identifying the key challenges that must be addressed to achieve practical and scalable quantum NLP solutions.

Bibliography

[1] Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: A learning algorithm for Boltzmann machines. Cognitive Science **9**(1), 147–169 (1985)

[2] Ahonen, H., Heinonen, O., Klemettinen, M., Verkamo, A.I.: Applying data mining techniques for descriptive phrase extraction in digital document collections. In: Proceedings of the IEEE International Forum on Research and Technology Advances in Digital Libraries (ADL '98), pp. 2–11. IEEE (1998)

[3] Aishwarya, C., Venkatesan, M., Prabhavathy, P.: Research oriented reviewing of quantum machine learning. International Research Journal on Advanced Science Hub **5**(05), 165–177 (2023)

[4] Aleksandrowicz, G., et al.: Qiskit: An open-source framework for quantum computing (2021)

[5] Alshawi, H.: The core language engine. MIT Press (1992)

[6] Anand, A., Romero, J., Degroote, M., Aspuru-Guzik, A.: Noise robustness and experimental demonstration of a quantum generative adversarial network for continuous distributions. Advanced Quantum Technologies **4**(4), 2000069 (2021)

[7] Andreev, N.D.: The intermediary language as the focal point of machine translation. In: Booth, A.D. (ed.) Machine translation, pp. 3–27. North Holland Publishing Company (1967)

[8] Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Sakkis, G., Spyropoulos, C.D., Stamatopoulos, P.: Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. arXiv preprint arXiv:cs/0009009 (2000)

[9] Arkin, E., Yadikar, N., Xu, X., Aysa, A., Ubul, K.: A survey: Object detection methods from CNN to transformer. Multimedia Tools and Applications **82**(14), 21353–21383 (2023)

[10] Arute, F., et al.: Quantum supremacy using a programmable superconducting processor. Nature **574**(7779), 505–510 (2019)

[11] Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: International Conference on Learning Representations (ICLR) (2015)

[12] Bangalore, S., Rambow, O., Whittaker, S.: Evaluation metrics for generation. In: Proceedings of the First International Conference on Natural Language Generation, vol. 14, pp. 1–8 (2000)

[13] Baud, R.H., Rassinoux, A.M., Scherrer, J.R.: Knowledge representation of discharge summaries. In: AIME 91, pp. 173–182. Springer (1991)

[14] Baud, R.H., Rassinoux, A.M., Scherrer, J.R.: Natural language processing and semantical representation of medical texts. Methods of Information in Medicine **31**(2), 117–125 (1992)

[15] Bausch, J.: Classifying data using near-term quantum devices. International Journal of Quantum Information **16**(08), 1840014 (2018)

[16] Bausch, J.: Recurrent quantum neural networks. arXiv preprint arXiv:2006.14619 (2020)

[17] Belhadef, H., Benchiheb, H., Lebdjiri, L.: Exploring the capabilities and limitations of VQC and QSVC for sentiment analysis on real-world and synthetic datasets. In: European Conference on Advances in Databases and Information Systems, pp. 415–424. Springer (2023)

[18] Benedetti, M., Garcia-Pintos, D., Perdomo, O., et al.: A generative modeling approach for benchmarking and training shallow quantum circuits. npj Quantum Information **5**(1), 45 (2019)

[19] Bengio, Y., Ducharme, R., Vincent, P.: A neural probabilistic language model. In: Proceedings of NIPS (2001)

[20] Benioff, P.: The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. Journal of Statistical Physics **22**(5), 563–591 (1980)

[21] Benson, E., Haghighi, A., Barzilay, R.: Event discovery in social media feeds. In: Proceedings of the 49th Annual Meeting of the ACL, vol. 1, pp. 389–398 (2011)

[22] Berger, A.L., Della Pietra, S.A., Della Pietra, V.J.: A maximum entropy approach to natural language processing. Computational Linguistics **22**(1), 39–71 (1996)

[23] Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., Lloyd, S.: Quantum machine learning. Nature **549**(7671), 195–202 (2017)

[24] Blanzieri, E., Bryl, A.: A survey of learning-based techniques of email spam filtering. Artificial Intelligence Review **29**(1), 63–92 (2008)

[25] Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017)

[26] Bondale, N., Maloor, P., Vaidyanathan, A., et al.: Extraction of information from open-ended questionnaires using natural language processing techniques. Computer Science and Informatics **29**(2), 15–22 (1999)

[27] Brassard, G., Høyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. Quantum Computation and Information, 53–74 (2002)

[28] Briscoe, E.J., Grover, C., Boguraev, B., Carroll, J.: A formalism and environment for the development of a large grammar of English. In: IJCAI 87, pp. 703–708 (1987)

[29] Carreras, X., Marquez, L.: Boosting trees for anti-spam email filtering. arXiv preprint arXiv:cs/0109015 (2001)

[30] Cerezo, M., Arrasmith, A., Babbush, R., Benjamin, S.C., Endo, S., Fujii, K., Coles, P.J.: Variational quantum algorithms. Nature Reviews Physics **3**(9), 625–644 (2021)

[31] Ceschini, A., Rosato, A., Panella, M.: Design of an LSTM cell on a quantum hardware. IEEE Transactions on Circuits and Systems II **69**(3), 1822–1826 (2022)

[32] Chalkidis, I., Fergadiotis, M., Malakasiotis, P., et al.: LEGAL-BERT: The muppets straight out of law school. arXiv preprint arXiv:2010.02559 (2020)

[33] Chen, B.Q., Niu, X.F.: Quantum neural network with improved quantum learning algorithm. International Journal of Theoretical Physics **59**(7), 1978–1991 (2020)

[34] Chen, Y., Li, F., Wang, J., et al.: Quantum recurrent encoder decoder neural network for performance trend prediction of rotating machinery. Knowledge-Based Systems **197**, 105863 (2020)

[35] Chen, S.Y.C.: Asynchronous training of quantum reinforcement learning. Procedia Computer Science **222**, 321–330 (2023)

[36] Chen, Z., Chen, J., Ding, G., Huang, H.: A lightweight CNN-based algorithm and implementation on embedded system for real-time face recognition. Multimedia Systems **29**(1), 129–138 (2023)

[37] Chiny, M., Chihab, M., Bencharef, O., Chihab, Y.: LSTM, VADER and TF-IDF based hybrid sentiment analysis model. International Journal of Advanced Computer Science and Applications **12**(7) (2021)

[38] Cho, K., Van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259 (2014)

[39] Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)

[40] Cohen, W.W.: Learning rules that classify e-mail. In: AAAI Spring Symposium on Machine Learning in Information Access, vol. 18, p. 25 (1996)

[41] Cohen, P.R., Morgan, J., Ramsay, A.M.: Intention in communication. American Journal of Psychology **104**(4) (2002)

[42] Collobert, R., Weston, J.: A unified architecture for natural language processing. In: Proceedings of the 25th ICML, pp. 160–167 (2008)

[43] Cong, I., Choi, S., Lukin, M.D.: Quantum convolutional neural networks. Nature Physics **15**(12), 1273–1278 (2019)

[44] Dave, K., Innan, N., Behera, B.K., Mumtaz, Z., Al-Kuwari, S., Farouk, A.: SentiQNF: A novel approach to sentiment analysis using quantum algorithms and neuro-fuzzy systems. arXiv preprint arXiv:2412.12731 (2024)

[45] Day, W., Chen, H.S., Sun, M.T.: QNet: A quantum-native sequence encoder architecture. In: 2023 IEEE International Conference on Quantum Computing and Engineering (QCE), vol. 1, pp. 246–255 (2023)

[46] Deutsch, D.: Quantum theory, the Church–Turing principle and the universal quantum computer. Proc. R. Soc. Lond. A **400**(1818), 97–117 (1985)

[47] Deutsch, D., Jozsa, R.: Rapid solution of problems by quantum computation. Proc. R. Soc. Lond. A **439**(1907), 553–558 (1992)

[48] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of NAACL-HLT 2019 (2019)

[49] Diab, M., Hacioglu, K., Jurafsky, D.: Automatic tagging of Arabic text: From raw text to base phrase chunks. In: Proceedings of HLT-NAACL 2004, pp. 149–152 (2004)

[50] Diep, D.N.: Some quantum neural networks. International Journal of Theoretical Physics **59**(7), 1179–1187 (2020)

[51] Doddington, G.: Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In: Proceedings of the 2nd HLT Research, pp. 138–145 (2002)

[52] Dong, Y., Fu, Y., Liu, H., et al.: An improved hybrid quantum-classical convolutional neural network for multi-class brain tumor MRI classification. Journal of Applied Physics **133**(6) (2023)

[53] Drucker, H., Wu, D., Vapnik, V.N.: Support vector machines for spam categorization. IEEE Transactions on Neural Networks **10**(5), 1048–1054 (1999)

[54] Dunlavy, D.M., O'Leary, D.P., Conroy, J.M., Schlesinger, J.D.: QCS: A system for querying, clustering and summarizing documents. Information Processing & Management **43**(6), 1588–1605 (2007)

[55] El-Din, D.M.: Enhancement bag-of-words model for solving the challenges of sentiment analysis. International Journal of Advanced Computer Science and Applications **7**(1) (2016)

[56] Emele, M.C., Dorna, M.: Ambiguity preserving machine translation using packed representations. In: Proceedings of the 36th ACL and 17th COLING, vol. 1, pp. 365–371 (1998)

[57] Encyclopedia.com: Physics: The Quantum Hypothesis, Scientific Thought: In Context. https://www.encyclopedia.com/science/science-magazines/physics-quantum-hypothesis (2020), last accessed 2026/05/15

[58] Feldman, S.: NLP meets the jabberwocky: Natural language processing in information retrieval. Online **23**, 62–73 (1999)

[59] Festa, P., Pardalos, P., Resende, M., Ribeiro, C.: Randomized heuristics for the Max-Cut problem. Optimization Methods and Software **17**(6), 1033–1058 (2002)

[60] Feynman, R.P.: Simulating physics with computers. International Journal of Theoretical Physics **21**(6), 467–488 (1982)

[61] Feynman, R.P.: Quantum mechanical computers. Optics News **11**(2), 11–20 (1985)

[62] Friedman, C., Cimino, J.J., Johnson, S.B.: A conceptual model for clinical radiology reports. In: Proceedings of the Annual Symposium on Computer Application in Medical Care, p. 829 (1993)

[63] Gao, T., Dontcheva, M., Adar, E., Liu, Z., Karahalios, K.: DataTone: Managing ambiguity in natural language interfaces for data visualization. In: Proceedings of the 28th Annual ACM UIST, pp. 489–500 (2015)

[64] Glasgow, B., et al.: MITA: An information-extraction approach to the analysis of free-form text in life insurance applications. AI Magazine **19**(1), 59 (1998)

[65] Gonçalves, C.P.: Quantum neural machine learning: Backpropagation and dynamics. NeuroQuantology **15**(1), 22–41 (2017)

[66] Gong, Y., Liu, X.: Generic text summarization using relevance measure and latent semantic analysis. In: Proceedings of the 24th Annual International ACM SIGIR, pp. 19–25 (2001)

[67] Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, vol. 27 (2014)

[68] Green Jr., B.F., Wolf, A.K., Chomsky, C., Laughery, K.: Baseball: An automatic question-answerer. In: Proceedings of the Western Joint IRE-AIEE-ACM Computer Conference, pp. 219–224 (1961)

[69] Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J.: LSTM: A search space odyssey. IEEE Trans. Neural Netw. Learn. Syst. 28(10), 2222–2232 (2017)

[70] Grishman, R., Sager, N., Raze, C., Bookchin, B.: The linguistic string parser. In: Proceedings of the National Computer Conference, pp. 427–434 (1973)

[71] Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the 28th Annual ACM STOC, pp. 212–219 (1996)

[72] Gyongyosi, L., Imre, S.: Training optimization for gate-model quantum neural networks. Scientific Reports **9**(1), 12665 (2019)

[73] Hassan, F., et al.: Performance evolution for sentiment classification using machine learning algorithm. Journal of Applied Research in Technology and Engineering **4**(2), 97–110 (2023)

[74] Havlíček, V., et al.: Supervised learning with quantum-enhanced feature spaces. Nature **567**(7747), 209–212 (2019)

[75] Hayes, P.J.: Intelligent high-volume text processing using shallow, domain-specific techniques. In: Text-based Intelligent Systems, pp. 227–242 (1992)

[76] Hendrix, G.G., Sacerdoti, E.D., Sagalowicz, D., Slocum, J.: Developing a natural language interface to complex data. ACM TODS **3**(2), 105–147 (1978)

[77] Hirschman, L., Grishman, R., Sager, N.: From text to structured information: Automatic processing of medical reports. In: Proceedings of the National Computer Conference, pp. 267–275 (1976)

[78] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)

[79] Hong, Y.Y., Arce, C.J.E., Huang, T.W.: A robust hybrid classical and quantum model for short-term wind speed forecasting. IEEE Access **11**, 90811–90824 (2023)

[80] Huang, Y., et al.: Quantum generative model with variable-depth circuit. Computers, Materials & Continua **65**(1), 445–458 (2020)

[81] Hur, T., Kim, L., Park, D.K.: Quantum convolutional neural network for classical data classification. Quantum Machine Intelligence **4**(1), 3 (2022)

[82] Hutchins, W.J.: Machine translation: Past, present, future. Ellis Horwood (1986)

[83] IBM: IBM Quantum Composer. https://quantum-computing.ibm.com/ (2021)

[84] Jia, K., et al.: Orthogonal deep neural networks. arXiv preprint arXiv:1905.05929 (2019)

[85] Jolliffe, I.T.: Principal component analysis. Springer (1986)

[86] Kamyab, M., Liu, G., Adjeisah, M.: Attention-based CNN and Bi-LSTM model based on TF-IDF and GloVe word embedding for sentiment analysis. Applied Sciences **11**(23), 11255 (2021)

[87] Kamp, H., Reyle, U.: Tense and aspect. In: From Discourse to Logic, pp. 483–689. Springer (1993)

[88] Kerenidis, I., Landman, J., Mathur, N.: Classical and quantum algorithms for orthogonal neural networks. arXiv preprint arXiv:2106.07198 (2021)

[89] Khan, T.M., Robles-Kelly, A.: Machine learning: Quantum vs classical. IEEE Access **8**, 219275–219294 (2020)

[90] Khan, L., Amjad, A., Afaq, K.M., Chang, H.T.: Deep sentiment analysis using CNN-LSTM architecture of English and Roman Urdu text shared in social media. Applied Sciences **12**(5), 2694 (2022)

[91] Kirilenko, A.P., Wang, L., Stepchenkova, S.O.: Sentiment analysis: Gaging opinions of large groups. In: Applied Data Science in Tourism, pp. 363–374. Springer (2022)

[92] Kockum, A.F., et al.: Lecture notes on quantum computing. arXiv preprint arXiv:2311.08445 (2023)

[93] Konar, D., et al.: Qutrit-inspired fully self-supervised shallow quantum learning network for brain tumor segmentation. IEEE Transactions on Neural Networks and Learning Systems (2021)

[94] LaBorde, M.L., Rogers, A.C., Dowling, J.P.: Finding broken gates in quantum circuits: Exploiting hybrid machine learning. Quantum Information Processing **19**(8), 241 (2020)

[95] Lass, R.: Phonology: An introduction to basic concepts. Cambridge University Press (1998)

[96] Lewis, D.D.: Naive (Bayes) at forty: The independence assumption in information retrieval. In: European Conference on Machine Learning, pp. 4–15. Springer (1998)

[97] Li, H.: Deep learning for natural language processing: Advantages and challenges. National Science Review **5**(1), 24–26 (2018)

[98] Li, Y., et al.: A quantum deep convolutional neural network for image recognition. Quantum Science and Technology **5**(4), 044003 (2020)

[99] Liddy, E.D.: Natural language processing. Encyclopedia of Information Science and Technology (2001)

[100] Lin, L.: Lecture notes on quantum algorithms for scientific computation. arXiv preprint arXiv:2201.08309 (2022)

[101] Liu, B.: Sentiment analysis: Mining opinions, sentiments, and emotions. Cambridge University Press (2020)

[102] Liu, W., et al.: A hybrid quantum-classical conditional generative adversarial network algorithm for human-centered paradigm in cloud. Eurasip Journal on Wireless Communications and Networking **2021**(1), 51 (2021)

[103] Lukac, M., Abdiyeva, K., Kameyama, M.: CNOT-measure quantum neural networks. In: Proceedings of the ISMVL, pp. 186–191. IEEE (2018)

[104] Luong, M.T., et al.: Addressing the rare word problem in neural machine translation. arXiv preprint arXiv:1410.8206 (2014)

[105] Lyman, M., et al.: Computer-structured narrative in ambulatory care. In: Proceedings of the Annual Symposium on Computer Application in Medical Care, p. 82 (1985)

[106] Mac-Kay Cisternas, C.A.: Quantum machine learning for predictive maintenance (2023)

[107] Mani, I., Maybury, M.T. (eds.): Advances in automatic text summarization, vol. 293. MIT Press (1999)

[108] Manin, Y.: Computable and uncomputable. Sovetskoye Radio (1980)

[109] Manning, C.D., Schütze, H.: Foundations of statistical natural language processing. MIT Press (1999)

[110] Martín, G.S., Droguett, E.L.: Quantum machine learning for health state diagnosis and prognostics (2021)

[111] McCallum, A., Nigam, K.: A comparison of event models for naive Bayes text classification. In: AAAI-98 Workshop on Learning for Text Categorization, vol. 752, pp. 41–48 (1998)

[112] McClean, J.R., et al.: Power of data in quantum machine learning. Nature Communications **12**(1), 2359 (2021)

[113] McCray, A.T., Srinivasan, S., Browne, A.C.: Lexical methods for managing variation in biomedical terminologies. In: Proceedings of the Annual Symposium on Computer Application in Medical Care, p. 235 (1994)

[114] McDonald, R., Crammer, K., Pereira, F.: Flexible text segmentation with structured multilabel classification. In: Proceedings of the HLT and EMNLP, pp. 987–994 (2005)

[115] McKeown, K.R.: Text generation. Cambridge University Press (1985)

[116] Meedinti, G.N., Srirekha, K.S., Delhibabu, R.: A quantum convolutional neural network approach for object detection and classification. arXiv preprint arXiv:2307.08204 (2023)

[117] Metropolis, N., Ulam, S.: The Monte Carlo method. Journal of the American Statistical Association **44**(247), 335–341 (1949)

[118] Mikolov, T., et al.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems (2013)

[119] Mohammad, S.M.: Sentiment analysis: Detecting valence, emotions, and other affectual states from text. In: Emotion Measurement, pp. 201–237. Woodhead Publishing (2016)

[120] Moll, N., et al.: Quantum optimization using variational algorithms on near-term quantum devices. Quantum Science and Technology **3**, 030503 (2018)

[121] Moré, J.J.: The Levenberg-Marquardt algorithm: Implementation and theory. In: Numerical Analysis, pp. 105–116. Springer (1978)

[122] Morel-Guillemaz, A.M., Baud, R.H., Scherrer, J.R.: Proximity processing of medical text. In: Medical Informatics Europe'90, pp. 625–630. Springer (1990)

[123] Morin, E.: Automatic acquisition of semantic relations between terms from technical corpora. In: Proceedings of the 5th TKE'99 (1999)

[124] Natural Language Processing RSS: Natural Language Processing. http://www.nlp.rss (2017), last accessed 2026/05/15

[125] Newatia, R.: Sentence classification using convolutional neural networks. Medium (2019)

[126] Nhan, N.T., et al.: A medical language processor for two Indo-European languages. In: Proceedings of the Annual Symposium on Computer Applications in Medical Care, pp. 554–558 (1989)

[127] Nicolay, D., Carletti, T.: Quantum neural networks achieving quantum algorithms. Communications in Computer and Information Science **830**, 3–15 (2018)

[128] Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information. Cambridge University Press (2000)

[129] Nieben, S., Och, F.J., Leusch, G., Ney, H.: An evaluation tool for machine translation: Fast evaluation for MT research. In: LREC 2000 (2000)

[130] Otter, D.W., Medina, J.R., Kalita, J.K.: A survey of the usages of deep learning for natural language processing. IEEE Transactions on Neural Networks and Learning Systems **32**(2), 604–624 (2020)

[131] Ouamane, N.E.H., Belhadef, H.: Deep reinforcement learning applied to NLP: a brief survey. In: 2022 2nd International Conference on New Technologies of Information and Communication (NTIC), pp. 1–5. IEEE (2022)

[132] Ouamane, N.E.H., Belhadef, H.: Proposed model for QCNN-based sentimental short sentences classification. In: International Conference of Reliable Information and Communication Technology, pp. 214–223. Springer (2023)

[133] Ouamane, N.E.H., Belhadef, H., Haddad, M.: Sentiment analysis of movie reviews based on quantum convolutional neural networks. The Journal of Supercomputing **81**(16), 1–20 (2025)

[134] Ovalle-Magallanes, E., et al.: Quantum angle encoding with learnable rotation applied to quantum–classical convolutional neural networks. Applied Soft Computing **141**, 110307 (2023)

[135] Ouyang, Y., Li, W., Li, S., Lu, Q.: Applying regression models to query-focused multi-document summarization. Information Processing & Management **47**(2), 227–237 (2011)

[136] Palmer, M., Gildea, D., Kingsbury, P.: The proposition bank: An annotated corpus of semantic roles. Computational Linguistics **31**(1), 71–106 (2005)

[137] Pandey, S., et al.: Quantum machine learning for natural language processing application. Physica A **627**, 129123 (2023)

[138] Papineni, K., et al.: BLEU: A method for automatic evaluation of machine translation. In: Proceedings of the 40th ACL, pp. 311–318 (2002)

[139] Pednault, E., et al.: Leveraging secondary storage to simulate deep 54-qubit Sycamore circuits. arXiv preprint arXiv:1910.09534 (2019)

[140] Pennington, J., Socher, R., Manning, C.D.: GloVe: Global vectors for word representation. In: Proceedings of EMNLP, pp. 1532–1543 (2014)

[141] Peters, M.E., et al.: Deep contextualized word representations. In: Proceedings of NAACL-HLT (2018)

[142] Porter, M.F.: An algorithm for suffix stripping. Program **14**(3), 130–137 (1980)

[143] Preskill, J.: Quantum computing in the NISQ era and beyond. Quantum **2**, 79 (2018)

[144] Ranjan, P., Basu, H.V.S.S.A.: Part of speech tagging and local word grouping techniques for natural language parsing in Hindi. In: Proceedings of the 1st ICON (2003)

[145] Rassinoux, A.M., et al.: Analysis of medical jargon: The RECIT system. In: AIME 95, pp. 42–52. Springer (1995)

[146] Rassinoux, A.M., et al.: Natural language processing of medical texts within the HELIOS environment. Computer Methods and Programs in Biomedicine **45**, S79–S96 (1994)

[147] Rennie, J.: ifile: An application of machine learning to e-mail filtering. In: Proc. KDD 2000 Workshop on Text Mining (2000)

[148] Ritter, A., Clark, S., Etzioni, O.: Named entity recognition in tweets: An experimental study. In: Proceedings of EMNLP, pp. 1524–1534 (2011)

[149] Rospocher, M., et al.: Building event-centric knowledge graphs from news. Web Semantics (2016)

[150] Sager, N., et al.: Medical language processing: Applications to patient data representation and automatic encoding. Methods of Information in Medicine **34**(1–2), 140–146 (1995)

[151] Sahami, M., et al.: A Bayesian approach to filtering junk e-mail. In: Learning for Text Categorization, vol. 62, pp. 98–105 (1998)

[152] Sakkis, G., Androutsopoulos, I.: Stacking classifiers for anti-spam filtering of e-mail. arXiv preprint arXiv:cs/0106040 (2001)

[153] Sakkis, G., et al.: A memory-based approach to anti-spam filtering for mailing lists. Information Retrieval **6**, 49–73 (2003)

[154] Santoro, A., et al.: Relational recurrent neural networks. Advances in Neural Information Processing Systems **31** (2018)

[155] Scherrer, J.R., et al.: Medical office automation integrated into the distributed architecture of a hospital information system. Methods of Information in Medicine **33**(2), 174–179 (1994)

[156] Schuld, M., Killoran, N.: Quantum machine learning in feature Hilbert spaces. Physical Review Letters **122**(4), 040504 (2019)

[157] Schuld, M., et al.: Circuit-centric quantum classifiers. Physical Review A **101**(3), 032308 (2020)

[158] Schuld, M., Petruccione, F.: Machine Learning with Quantum Computers, vol. 676. Springer (2021)

[159] Seal, D., Roy, U.K., Basak, R.: Sentence-level emotion detection from text based on semantic rules. In: Information and Communication Technology for Sustainable Development, vol. 933. Springer (2020)

[160] Sha, F., Pereira, F.: Shallow parsing with conditional random fields. In: Proceedings of NAACL-HLT, vol. 1, pp. 134–141 (2003)

[161] Shankar, R.: Principles of quantum mechanics. Plenum Press (1994)

[162] Sharma, S., Srinivas, P.Y.K.L., Balabantaray, R.C.: Emotion detection using online machine learning method and TLBO on mixed script. In: Proceedings of LREC 2016, pp. 47–51 (2016)

[163] Sharma, A.K., Chaurasia, S., Srivastava, D.K.: Sentimental short sentences classification by using CNN deep learning model with fine tuned Word2Vec. Procedia Computer Science **167**, 1139–1147 (2020)

[164] Shemtov, H.: Ambiguity management in natural language generation. PhD thesis, Stanford University (1997)

[165] Simon, D.: On the power of quantum computation. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science, pp. 116–123. IEEE (1994)

[166] Singh, J., Sharma, G.: Sentiment analysis study of human thoughts using machine learning techniques. In: 2023 International Conference on Disruptive Technologies (ICDT), pp. 776–785. IEEE (2023)

[167] Situ, H., et al.: Quantum generative adversarial network for generating discrete distribution. Information Sciences **538**, 193–208 (2020)

[168] Skavysh, V., et al.: Quantum Monte Carlo for economics: Stress testing and macroeconomic deep learning. Journal of Economic Dynamics and Control **153**, 104680 (2023)

[169] Skolik, A., et al.: Layerwise learning for quantum neural networks. Quantum Machine Intelligence **3**(1), 5 (2021)

[170] Small, S.L., Cortell, G.W., Tanenhaus, M.K.: Lexical ambiguity resolutions. Morgan Kaufmann (1988)

[171] Socher, R., et al.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of EMNLP, pp. 1631–1642 (2013)

[172] Srikumar, M., Hill, C.D., Hollenberg, L.C.L.: Clustering and enhanced classification using a hybrid quantum autoencoder. Quantum Science and Technology **7**(1), 015020 (2021)

[173] Strubell, E.: An introduction to quantum algorithms. COS498 Chawathe Spring **13**, 19 (2011)

[174] Sun, X., et al.: Modeling latent-dynamic in shallow parsing. In: Proceedings of COLING, vol. 1, pp. 841–848 (2008)

[175] Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems (2014)

[176] Suzuki, Y., et al.: Natural quantum reservoir computing for temporal information processing. arXiv preprint arXiv:2107.05808 (2021)

[177] Taboada, M.: Sentiment analysis: An overview from linguistics. Annual Review of Linguistics **2**(1), 325–347 (2016)

[178] Tacchino, F., Macchiavello, C., Gerace, D., Bajoni, D.: An artificial neuron implemented on an actual quantum processor. npj Quantum Information **5**(1), 1–8 (2019)

[179] Tacchino, F., et al.: Quantum implementation of an artificial feed-forward neural network. Quantum Science and Technology **5**(4), 044003 (2020)

[180] Tacchino, F., et al.: Variational learning for quantum artificial neural networks. IEEE Transactions on Quantum Engineering **2**, 1–10 (2021)

[181] Tan, K.L., et al.: RoBERTa-LSTM: A hybrid model for sentiment analysis with transformers and recurrent neural network. IEEE Access (2022)

[182] Tapaswi, N., Jain, S.: Treebank based deep grammar acquisition and part-of-speech tagging for Sanskrit sentences. In: Proceedings of CONSEG 2012. IEEE (2012)

[183] Thomas, C.: Recurrent neural networks and natural language processing. Towards Data Science (2019)

[184] Tillmann, C., et al.: Accelerated DP based search for statistical translation. In: Eurospeech (1997)

[185] Tychola, K.A., Kalampokas, T., Papakostas, G.A.: Quantum machine learning—an overview. Electronics **12**(11), 2379 (2023)

[186] Umber, A., Bajwa, I.: Minimizing ambiguity in natural language software requirements specification. In: 6th IC DIM, pp. 102–107 (2011)

[187] Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)

[188] Vidal, G.: Class of quantum many-body states that can be efficiently simulated. Physical Review Letters **101**(11), 110501 (2008)

[189] Wahlster, W., Kobsa, A.: User models in dialog systems. In: User Models in Dialog Systems, pp. 4–34. Springer (1989)

[190] Walton, D.: A pragmatic synthesis. In: Fallacies Arising from Ambiguity, vol. 1. Springer (1996)

[191] Wan, X.: Using only cross-document relationships for both generic and topic-focused multi-document summarizations. Information Retrieval **11**(1), 25–49 (2008)

[192] Wang, W., Gang, J.: Application of convolutional neural network in natural language processing. In: Proceedings of ICISCAE, pp. 64–70. IEEE (2018)

[193] Wang, J., et al.: Orthogonal convolutional neural networks. arXiv preprint arXiv:1911.12207 (2020)

[194] Wang, H., et al.: A quantum approximate optimization algorithm with metalearning for maxcut problem. Mathematical Problems in Engineering **2021**, 6655455 (2021)

[195] Wiese, G., Weissenborn, D., Neves, M.: Neural domain adaptation for biomedical question answering. arXiv preprint arXiv:1706.03610 (2017)

[196] Woods, W.A.: Semantics and quantification in natural language question answering. Advances in Computers **17**, 1–87 (1978)

[197] Wu, Y., et al.: Strong quantum computational advantage using a superconducting quantum processor. Physical Review Letters **127**(18), 180501 (2021)

[198] Xia, T.: A constant time complexity spam detection algorithm for boosting throughput on rule-based filtering systems. IEEE Access **8**, 82653–82661 (2020)

[199] Yi, J., et al.: Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In: Proceedings of the Third IEEE ICDM, pp. 427–434 (2003)

[200] Yong, L., et al.: Closing the "quantum supremacy" gap: Achieving real-time simulation of a random quantum circuit using a new Sunway supercomputer. In: Proceedings of SC21, pp. 1–12 (2021)

[201] Yu, S., et al.: A multi-stage memory augmented neural network for machine reading comprehension. Proceedings of the Workshop on Machine Reading for Question Answering (2018)

[202] Yuan, F., Zhang, Z., Fang, Z.: An effective CNN and Transformer complementary network for medical image segmentation. Pattern Recognition **136**, 109228 (2023)

[203] Zeguendry, A., Jarir, Z., Quafafou, M.: Quantum convolutional neural network for classical data classification. Entropy **25**, 2–287 (2023)

[204] Zeroual, I., Lakhouaja, A., Belahbib, R.: Towards a standard part of speech tagset for the Arabic language. Journal of King Saud University **29**(2), 171–178 (2017)

[205] Zhang, Y., Song, D., Li, X., Zhang, P.: Unsupervised sentiment analysis of Twitter posts using density matrix representation. In: Advances in Information Retrieval, pp. 316–329. Springer (2018)

[206] Zhang, Y., et al.: Quantum-inspired interactive networks for conversational sentiment analysis. arXiv preprint arXiv:1906.03550 (2019)

[207] Zhang, P., et al.: TextTN: Probabilistic encoding of language on tensor network. arXiv preprint arXiv:2107.03058 (2021)

[208] Zhong, H.S., et al.: Quantum computational advantage using photons. Science **370**(6523), 1460–1463 (2020)

[209] Zhu, Q., et al.: Quantum computational advantage via 60-qubit 24-cycle random circuit sampling. Science Bulletin **66**(23), 2413–2420 (2021)

[210] Zoufal, C., Lucchi, A., Woerner, S.: Variational quantum Boltzmann machines. Quantum Machine Intelligence **3**(1) (2021)