

Máster Universitario en Computación y Sistemas Inteligentes

Konputazioari eta Sistema Adimendunei buruzko Unibertsitate Masterra

Proyecto fin de máster

Master amaierako proiektua

Exploración de enfoques cuánticos para su aplicación
en tareas de generación de lenguaje natural

Iker Marcelo Pérez

Director: Mikel Emaldi Manrique

Bilbao, junio de 2025

Máster Universitario en Computación y Sistemas Inteligentes

Konputazioari eta Sistema Adimendunei buruzko Unibertsitate Masterra

Proyecto fin de máster

Master amaierako proiektua

Exploración de enfoques cuánticos para su aplicación
en tareas de generación de lenguaje natural

Iker Marcelo Pérez

Director: Mikel Emaldi Manrique

Bilbao, junio de 2025

Resumen

El procesamiento cuántico del lenguaje natural es la convergencia entre el procesamiento del lenguaje natural clásico y las tecnologías cuánticas emergentes. Sobre la teoría, el procesamiento cuántico del lenguaje natural busca aprovechar propiedades de la computación cuántica para modelar, procesar y generar lenguaje natural de manera más eficaz que los métodos clásicos.

El objetivo de este Proyecto Fin de Máster es estudiar el estado del arte actual de la computación cuántica y su aplicación en tareas de generación de lenguaje natural. Para cumplir este objetivo, se implementarán distintos enfoques cuánticos mediante frameworks como Qiskit y PennyLane, ejecutándose tanto en simuladores como en ordenadores cuánticos reales a través de IBM Quantum. Estas implementaciones permitirán evaluar la capacidad de los distintos enfoques cuánticos para generar lenguaje natural. Los resultados se validarán y se compararán con enfoques clásicos, como redes neuronales generativas simples, implementadas específicamente para replicar las mismas tareas que los enfoques cuánticos.

Además de comprobar su viabilidad, el proyecto explorará si los enfoques cuánticos permiten representar dependencias semánticas de forma más compacta o eficaz, o si ofrecen mejoras potenciales en tareas específicas. Al final del proyecto se espera determinar si existe una ventaja cuántica en la generación de lenguaje natural, identificando las limitaciones actuales y las posibles mejoras a futuro.

Descriptores

Procesamiento de Lenguaje Natural Cuántico, Modelos Generativos Cuánticos, Computación Cuántica Variacional, Aprendizaje Cuántico Híbrido

Índice

1. Introducción	1
2. Objetivos y Alcance	2
2.1. Objetivos	2
2.2. Alcance	2
3. Antecedentes	3
3.1. Procesamiento del Lenguaje Natural	3
3.2. Fundamentos de Computación Cuántica	4
3.2.1. Qubit	4
3.2.2. Superposición y Medición Cuántica	5
3.2.3. Entrelazamiento	6
3.2.4. Circuitos Cuánticos y Puertas Lógicas	7
4. Estado del arte	10
4.1. La era NISQ	10
4.2. Quantum Natural Language Processing	11
4.2.1. DisCoCat	11
4.2.2. QSANN	13
4.2.3. QRNN	13
4.3. Modelos generativos cuánticos	14
4.3.1. Q1Synth	14
4.3.2. Basado en clasificación	14
4.3.3. QGANs	16
4.3.4. QCBMs	18
4.3.5. QBMs	19
5. Desarrollo del proyecto	20
5.1. Quantum Generative Adversarial Network	20
5.1.1. Clasificador cuántico	20
5.1.2. Discriminador cuántico	25

5.1.3.	Generador y Quantum GAN	26
5.2.	Generación de lenguaje mediante Quantum Circuit Born Machines	27
5.2.1.	Generar caracteres con 2 qubits	28
5.2.2.	Generar secuencias de 2 caracteres con 4 qubits	30
5.2.3.	Generar secuencias de 3 palabras con 9 qubits	32
5.2.4.	Conclusiones y perspectivas del enfoque	35
5.3.	Generación de secuencias mediante Simulated Annealing y SWAP Test	37
5.3.1.	Codificación con bitstrings	37
5.3.2.	Codificación por fase	40
5.3.3.	Codificación por amplitud y fase con FastText	43
5.4.	Transformer con auto-atención cuántica	44
5.4.1.	Primera versión	45
5.4.2.	Segunda versión	48
5.4.3.	Tercera versión	57
5.5.	Quantum Long Short-Term Memory	58
5.5.1.	Conjunto de datos y preprocesamiento	58
5.5.2.	LSTM clásico	58
5.5.3.	QLSTM híbrido	59
5.5.4.	Entrenamiento y Resultados	60
5.5.5.	Búsqueda de arquitectura cuántica	62
5.5.6.	QLSTM con cuatro puertas cuánticas y re-uploading	63
6.	Valoración ética del proyecto	67
6.1.	Metodología de análisis ético	67
6.2.	Cuestiones éticas relevantes	67
6.3.	Evaluación de impactos y medidas de mitigación	67
6.4.	Conclusiones	68
7.	Plan de trabajo	69
7.1.	Fases	69
7.2.	Tareas	70
7.3.	Diagrama de Gantt	70

8. Presupuesto	72
9. Estructura del proyecto	73
9.1. Estructura del repositorio	73
10. Conclusiones y líneas futuras	74
10.1. Conclusiones	74
10.2. Líneas Futuras	75

Índice de figuras

1.	Visualización de un qubit en estado 0 mediante la esfera de Bloch. [1]	5
2.	Visualización de un qubit en superposición mediante la esfera de Bloch. [1]	5
3.	Paradoja del gato de Schrödinger [2]	6
4.	Circuito cuántico estado de Bell [3]	9
5.	Diagrama de cadenas utilizado como ejemplo en [4]	12
6.	Oración “Alice generates language” convertida a circuito cuántico [4]	12
7.	Representación gráfica del algoritmo descrito [4].	16
8.	Número de titulares por categoría	20
9.	Resultados clasificador clásico	20
10.	Ejemplos de los diagramas creados con Lambeq	21
11.	Tipos atómicos encontrados	22
12.	Circuito cuántico generado por Lambeq para una de las oraciones del dataset.	23
13.	Entrenamiento del clasificador.	24
14.	Enter Caption	24
15.	Ejemplo de frases aceptas y rechazadas por BobcatParser	25
16.	Entrenamiento del discriminador.	25
17.	Entrenamiento de la QGAN	27
18.	Circuito cuántico para generar caracteres con 2 qubits.	28
19.	Circuito cuántico para generar cadenas de 2 caracteres.	31
20.	Circuito cuántico para generar secuencias de 3 palabras.	34
21.	Grafo de cómputo generado con torchviz para la red neuronal clásica.	36
22.	Diagrama del circuito que implementa el SWAP test con los registros candidato y referencia. Codificación por bitstrings.	38
23.	Visualización del vocabulario codificado por fase en la esfera de bloch.	40
24.	Diagrama del circuito que implementa el SWAP test con los registros candidato y referencia. Codificación por fase.	41
25.	Diagrama basado en SWAP test para secuencias de 5 palabras	42
26.	Visualización de 8 palabras codificadas por amplitud y fase según la similitud en FastText.	44
27.	Circuito de auto-atención cuántica de 8 dimensiones replicado en PennyLane.	46
28.	Topología y errores del dispositivo ibm_sherbrooke [5].	50

29.	Historial de ejecuciones de la primera prueba[5].	51
30.	Resultados completos de la ejecución. Visualizado en Google Colab.	52
31.	Resultado de la segunda ejecución	53
32.	Historial de ejecuciones de la segunda prueba[5].	54
33.	Resultado de la tercera ejecución	55
34.	Historial de ejecuciones de la tercera prueba	56
35.	Enter Caption	57
36.	Arquitectura interna de una celda LSTM.	59
37.	Circuito cuántico usado como capa dentro del modelo híbrido.	60
38.	Gráfica de la curva de pérdida de la LSTM y la QLSTM	61
39.	Accuracy alcanzada por los modelos	61
40.	Resultados del Grid Search para la QLSTM	62
41.	Circuito cuántico usado como capa dentro del modelo híbrido con 5 qubits y 2 capas de profundidad	62
42.	Gráfica de la curva de pérdida de la LSTM y la QLSTM con 5 qubits y 2 capas de profundidad	63
43.	Gráfica de la curva de pérdida de la LSTM y la QLSTM con 4 circuitos cuánticos	64
44.	Gráfica de la curva de pérdida para dataset con 50 frases.	65
45.	Accuracy para dataset con 50 frases.	65
46.	Diagrama de Gantt parte 1	71
47.	Diagrama de Gantt parte 2	71
48.	Diagrama de Gantt parte 3	71

Índice de tablas

1.	Asignación de etiquetas y distribución de probabilidad.	28
2.	Distribución generada para diferentes valores de épocas.	29
3.	Codificación de secuencias de 2 caracteres a partir de bitstrings de 4 bits.	30
4.	Comparación entre la distribución generada y la distribución objetivo para secuencias de 2 caracteres.	32
5.	Codificación de palabras con bitstrings de 3 bits.	32
6.	Distribución objetivo de probabilidad para secuencias de 3 palabras.	33
7.	Comparación entre la distribución generada y la distribución objetivo para las secuencias relevantes de 3 palabras.	35
8.	Comparación entre la distribución generada por el modelo cuántico, el modelo clásico y la distribución objetivo.	35
9.	Codificación del vocabulario mediante ángulos en radianes.	40
10.	Resultados de texto generado a partir de distintas frases de entrada.	48
11.	Resumen del número de parámetros por componente de los modelos.	62
12.	Categorías léxicas usadas para la construcción del dataset.	64
13.	Comparativa de accuracy entre LSTM y QLSTM en 5 repeticiones con distintas semillas.	66
14.	Comparativa de parámetros entre el modelo LSTM Generator y el modelo QLSTM Gated Generator.	66
15.	Listado de tareas del proyecto organizadas por fase	70
16.	Resumen de horas y costes por perfil profesional.	72
17.	Costes asociados al material y software utilizado.	72
18.	Resumen del presupuesto del proyecto	72

1. Introducción

En los últimos años, el Procesamiento del Lenguaje Natural ha logrado consolidarse como una de las áreas con mayor crecimiento de la inteligencia artificial, impulsada en gran parte por el aumento en la disponibilidad de datos textuales y la progresión de las redes neuronales profundas.

No obstante, los progresos en NLP se enfrentan a desafíos cada vez más complejos. La comprensión del contexto, la ambigüedad del lenguaje humano y la gestión de grandes cantidades de datos, demandan un aumento continuo en recursos computacionales y técnicas de optimización.

Al mismo tiempo, emerge la computación cuántica como nuevo campo de estudio y crece su interés debido a su capacidad de enfrentar problemas de alta complejidad haciendo uso de principios de la mecánica cuántica, como la superposición y el entrelazamiento.

En este contexto, el Procesamiento Cuántico del Lenguaje Natural se propone como la fusión entre el NLP clásico y las emergentes tecnologías cuánticas. La idea es modelar palabras, frases y relaciones semánticas como estados cuánticos con el objetivo de conseguir una ventaja cuántica. Aunque es cierto que a día de hoy los ordenadores cuánticos cuentan con un número limitado de qubits y presentan errores significativos en términos de “ruido”, ya se han llevado a cabo varias demostraciones experimentales y marcos teóricos que señalan la capacidad del QNLP para cambiar la forma en la que procesamos y generamos lenguaje.

2. Objetivos y Alcance

En el capítulo de “Objetivos y Alcance” se establecen de forma clara las metas que se desean conquistar y el perímetro dentro del cual operará este Proyecto Fin de Máster. Se comenzará desglosando tanto el objetivo principal del proyecto como aquellos secundarios y, posteriormente, se aclarará el alcance del proyecto junto con sus limitaciones.

2.1. Objetivos

El Proyecto Fin de Máster tiene como objetivo principal estudiar la viabilidad de aplicar computación cuántica en tareas de NLP generativo y determinar si esta aplicación supone alguna ventaja frente a los métodos clásicos.

Además, se definen los siguientes objetivos secundarios:

1. Realizar una revisión actualizada del estado del arte en Computación Cuántica, especialmente aplicada a tareas de NLP y tareas generativas.
2. Diseñar e implementar prototipos de enfoques cuánticos utilizando frameworks de computación cuántica como Qiskit y PennyLane.
3. Validar los resultados de los enfoques y modelos cuánticos y compararlos con modelos de NLP clásicos.
4. Sintetizar conclusiones y proponer líneas de investigación a corto y medio plazo.
5. Evaluar el impacto ético del proyecto.

2.2. Alcance

Este proyecto se sitúa en la intersección entre la lingüística computacional y la computación cuántica y, dado el estado incipiente de esta última, este apartado fija las fronteras dentro de las cuales se trabajará. El alcance y las limitaciones del proyecto incluyen los siguientes aspectos fundamentales:

1. Los circuitos y modelos se implementados se ejecutarán en simuladores de Qiskit o PennyLane, pudiendo ejecutarse también en hardware real de IBM Quantum cuando sea viable y se considere que dichas ejecuciones aporten algo diferente a las realizadas en simuladores.
2. Las ejecuciones en hardware real de IBM Quantum están limitadas a 10 minutos mensuales, debido a su plan gratuito.
3. Los casos de uso de los experimentos serán acotados, empleando corpus y vocabularios reducidos, debido a las limitaciones actuales en hardware cuántico. La extensión a vocabularios y dominios amplios queda fuera de alcance.
4. Se priorizarán diseños superficiales en los circuitos cuánticos, técnicas avanzadas de error-mitigation se considerarán trabajo futuro.

Esta delimitación asegura que los objetivos definidos sean abordables con los medios y plazos disponibles, evitando comprometer la validez de los resultados.

3. Antecedentes

3.1. Procesamiento del Lenguaje Natural

El Procesamiento del Lenguaje Natural es un campo interdisciplinario de la informática, la inteligencia artificial y la lingüística que se centra en la interacción entre los ordenadores y el lenguaje humano [6]. El NLP, abarca el desarrollo de modelos y algoritmos que permiten a las máquinas reconocer, comprender y generar información en lenguaje natural, ya sea por texto o voz, logrando así comunicarse con personas de manera eficaz [7]. Para alcanzar este objetivo, el NLP combina la lingüística computacional (que aporta reglas y estructuras del lenguaje) con técnicas estadísticas, aprendizaje automático y especialmente aprendizaje profundo, aprovechando la gran disponibilidad actual de datos textuales y el aumento en el poder de cómputo [7].

La evolución histórica del NLP ha estado marcada por cambios de enfoque técnico. Hasta los años 1980, la mayoría de los sistemas se basaban en extensos conjuntos de reglas lingüísticas definidas manualmente. Posteriormente, con el incremento del poder computacional y la disponibilidad de grandes corpus de texto, surgieron métodos estadísticos y de aprendizaje automático que iniciaron una revolución en el campo [6]. A partir de entonces, los sistemas comenzaron a aprender patrones y estructuras del lenguaje directamente a partir de los datos, en lugar de depender únicamente de reglas predefinidas. Ya en la última década, el aprendizaje profundo mediante redes neuronales ha impulsado un nuevo salto cualitativo, permitiendo captar mejor las sutilezas del lenguaje y manejar su complejidad de forma mucho más efectiva que las técnicas previas.

La comprensión automática del lenguaje conlleva la integración de múltiples niveles de análisis, desde el reconocimiento léxico y morfológico de las palabras, pasando por el análisis sintáctico de la estructura oracional, hasta la interpretación semántica y pragmática del mensaje en contexto [6]. Cada uno de estos niveles presenta importantes desafíos. Por ejemplo, la ambigüedad inherente del lenguaje humano hace que una misma palabra o frase pueda tener distintas interpretaciones según el contexto, lo que obliga a los modelos a desambiguar apoyándose en información contextual y conocimiento lingüístico [6]. Asimismo, captar relaciones gramaticales complejas y dependencias de largo alcance en un texto extenso es una tarea no trivial. Los primeros enfoques basados en modelos de lenguaje estadísticos o incluso las redes neuronales recurrentes iniciales tenían dificultades para considerar un contexto amplio, limitando la coherencia cuando debían procesar párrafos o diálogos extensos. Este panorama comenzó a cambiar con la introducción de la arquitectura Transformer, que revolucionó el NLP al incorporar mecanismos de auto-atención capaces de identificar las partes más relevantes de la secuencia de entrada y manejar eficientemente las dependencias a larga distancia, superando así las limitaciones de los modelos previos [8].

Dentro de las áreas del NLP, la generación de lenguaje natural ha cobrado especial relevancia en los últimos años. Tradicionalmente, la generación automática de texto se abordaba mediante reglas predefinidas o con modelos estadísticos de lenguaje, los cuales ofrecían resultados bastante limitados en fluidez y variedad. La llegada de los métodos neuronales de aprendizaje profundo amplió drásticamente estas capacidades. Los primeros modelos secuencia a secuencia con mecanismos de atención demostraron que era posible traducir frases entre idiomas o resumir documentos con una calidad muy superior a la de los enfoques clásicos. Posteriormente, la adopción generalizada de arquitecturas Transformer potenció aún más la calidad de la generación de texto, al punto de hacer factible el entrenamiento de modelos de lenguaje de tamaño sin precedentes, especializados en predecir la siguiente palabra de una secuencia y capaces de producir texto coherente y de alta calidad de forma autónoma [8].

Con estos avances surgieron los modelos de lenguaje de gran tamaño, entre los cuales destaca la familia GPT. Un ejemplo es GPT-3, un modelo con 175 mil millones de parámetros cuyo entrenamiento masivo le permitió aprender patrones lingüísticos complejos y generar texto notablemente coherente y contextualizado en una amplia gama de tareas de lenguaje [9]. Versiones más recientes como GPT-4 [10], han llegado aún más lejos, mostrando una fluidez y capacidad de adaptación al contexto cercanas a las de los humanos.

Estos avances han posicionado al NLP en el núcleo de la inteligencia artificial actual, impulsando una nueva ola de aplicaciones de IA generativa impensables hace unos años. Hoy en día, tecnologías cotidianas como los motores de búsqueda en internet, los asistentes conversacionales inteligentes, los sistemas de traducción automática en tiempo real o las herramientas de resumen de textos dependen críticamente de técnicas y modelos de NLP [7].

Sin embargo, la contraparte de estos logros es el coste computacional elevadísimo de entrenar y desplegar los modelos punteros, que a menudo cuentan con decenas de miles de millones de parámetros entrenables, o incluso más. Esta realidad ha comenzado a motivar la exploración de enfoques alternativos e innovadores para seguir mejorando el NLP de forma eficiente en el futuro.

3.2. Fundamentos de Computación Cuántica

Para entender la computación cuántica es esencial explicar algunos principios clave. Los conceptos presentados a continuación son aquellos que estudié al inicio del proyecto, ya que partía sin conocimientos previos en este campo. Esta revisión me permitió adquirir las bases necesarias para abordar con seguridad el desarrollo experimental posterior.

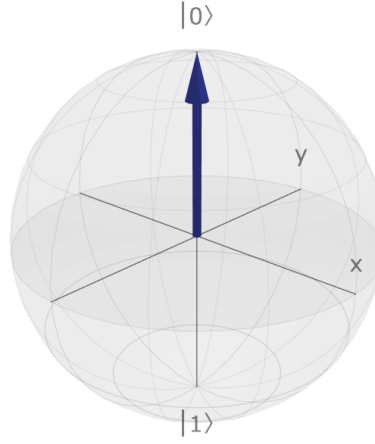
3.2.1. Qubit

Los ordenadores cuánticos, en lugar de utilizar bits utilizan qubits. Mientras que un bit solo puede tomar 2 valores, 0 o 1, un qubit puede encontrarse en un tercer estado conocido como superposición. Una superposición representa 0, 1 y todas las posiciones intermedias tomadas a la vez, para un total de tres posiciones distintas [11].

Un qubit se puede representar mediante la esfera de Bloch, donde cualquier estado puro de un qubit se puede representar como:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right) |0\rangle + e^{i\phi} \sin\left(\frac{\theta}{2}\right) |1\rangle,$$

con $0 \leq \theta \leq \pi$ y $0 \leq \phi < 2\pi$.

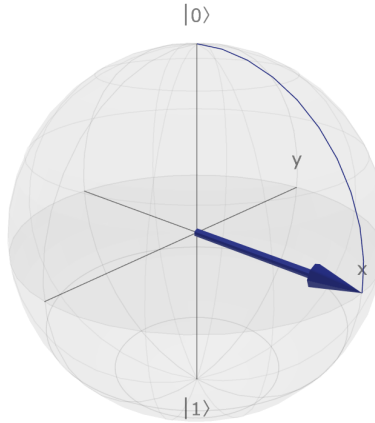


bloch.kherb.io

Figura 1: Visualización de un qubit en estado 0 mediante la esfera de Bloch. [1]

3.2.2. Superposición y Medición Cuántica

En términos computacionales, la superposición significa que un qubit, puede estar simultáneamente en una combinación de los estados $|0\rangle$ y $|1\rangle$.



bloch.kherb.io

Figura 2: Visualización de un qubit en superposición mediante la esfera de Bloch. [1]

En la Figura 2 se ha rotado el eje y en 90° para lograr que el vector del qubit, que inicialmente apuntaba hacia el eje $z(|0\rangle)$, ahora quede en el plano xz .

Matricialmente, una rotación de 90° en y se representa mediante la matriz $R_y(\pi/2)$:

$$R_y\left(\frac{\pi}{2}\right) = \begin{pmatrix} \cos\left(\frac{\pi}{4}\right) & -\sin\left(\frac{\pi}{4}\right) \\ \sin\left(\frac{\pi}{4}\right) & \cos\left(\frac{\pi}{4}\right) \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

Después de aplicar la rotación $R_y(90^\circ)$, el vector del qubit apunta hacia el plano xz , formando un ángulo de 45° respecto al eje z . Esto implica que el qubit se encuentra en una superposición equitativa entre $|0\rangle$

y $|1\rangle$, que puede escribirse como:

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

Cuando un estado cuántico es medido u observado, el estado cuántico se rompe y la superposición colapsa a un resultado concreto. Esta medición es probabilística, y la probabilidad de obtener un resultado depende de los coeficientes de la superposición.

En términos de qubits, si $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, la probabilidad de medir el estado $|0\rangle$ es $|\alpha|^2$ y de $|1\rangle$ es $|\beta|^2$.

Una forma útil, aunque exagerada, de entender los fenómenos de superposición y medición cuántica es el experimento mental del gato de Schrödinger [12]. Este experimento fue propuesto por Erwin Schrödinger en 1935, y describe un escenario donde, mientras la caja siga cerrada, el gato está vivo y muerto al mismo tiempo, es decir, se encuentra en una superposición de ambos estados. En el momento en que un observador abre la caja y observa al gato (realiza una medición) la superposición cuántica colapsa a un resultado clásico definitivo, el gato está vivo o muerto.

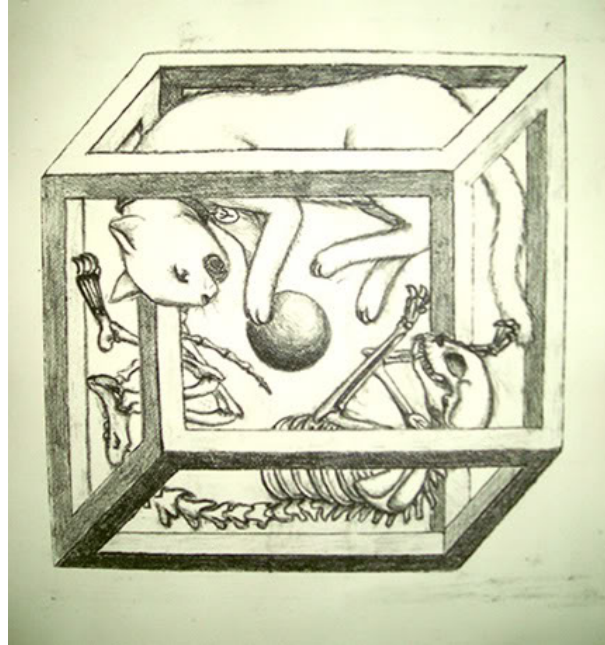


Figura 3: Paradoja del gato de Schrödinger [2]

Aunque la motivación original de Erwin Schrödinger era señalar lo absurdo de aplicar reglas cuánticas a sistemas macroscópicos, con el tiempo, debido a su popularidad y a lo impactante del planteamiento, se ha convertido en un ejemplo clásico que permite entender conceptos cuánticos que suelen resultar contraintuitivos, recurriendo a una situación tangible como la vida o la muerte de un gato.

3.2.3. Entrelazamiento

El entrelazamiento cuántico se da cuando dos o más partículas están correlacionadas de forma que el estado de cada partícula no puede describirse independientemente del estado de las demás, incluso cuando los objetos estén separados espacialmente.

Desde el punto de vista matemático, esto significa que el estado conjunto del sistema no puede escribirse como un producto de estados individuales. Por ejemplo, un sistema de dos qubits no entrelazados se puede expresar como:

$$|\psi\rangle = |\alpha\rangle_A \otimes |\beta\rangle_B$$

En cambio, si los qubits están entrelazados, ya no se puede descomponer el sistema en partes independientes. En lugar de eso, se representa mediante un estado global inseparable como:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

El entrelazamiento entre qubits permite realizar operaciones conjuntas muy potentes, donde la medición de un qubit afecta instantáneamente al estado del resto de qubits entrelazados, sin importar la distancia entre qubits.

Sin embargo, no es posible transmitir información clásica más rápido que la velocidad de la luz mediante el entrelazamiento, ya que se necesita un canal clásico para interpretar la correlación [13].

3.2.4. Circuitos Cuánticos y Puertas Lógicas

Un circuito cuántico es el modelo más común para realizar computación cuántica [14]. Su analogía en la computación clásica son los circuitos lógicos.

Los circuitos cuánticos suelen representarse con líneas horizontales, que representan a los qubits y encima de esas líneas se colocan cajas que representan puertas cuánticas. Se lee de izquierda a derecha, comenzando por los qubits iniciales. Estos qubits suelen inicializarse automáticamente en $|0\rangle$, y según avanzan por las distintas puertas cuánticas su estado va cambiando.

Las puertas cuánticas actúan sobre los qubits moldeando su estado, igual que las puertas clásicas con los bits. A diferencia de las puertas clásicas, las cuánticas se rigen por las leyes de la mecánica cuántica, lo que introduce algunos comportamientos asombrosos:

- **Unitariedad:** Cada puerta cuántica es una operación unitaria. En términos sencillos, eso significa que transforma el estado del sistema sin “perder” nada, la norma (o magnitud) del estado siempre se conserva.
- **Reversibilidad:** Siempre puedes deshacer una puerta cuántica. Si aplicas una operación U que transforma un estado inicial $|\psi\rangle$ en otro $|\phi\rangle$, la inversa U^\dagger siempre puede restaurar $|\psi\rangle$ a partir de $|\phi\rangle$.
- **Superposición:** Estas puertas pueden operar sobre qubits en estados combinados, permitiendo procesar múltiples posibilidades a la vez.
- **Entrelazamiento:** Algunas puertas pueden entrelazar qubits, creando conexiones profundas entre ellos, incluso si están separados físicamente.

Puertas Cuánticas de 1 Qubit

Estas puertas actúan sobre un único qubit, y aunque parezcan simples, abren la puerta a comportamientos asombrosos:

- **Puerta X:** Es el equivalente cuántico del NOT clásico. Si el qubit está en $|0\rangle$ lo cambia a $|1\rangle$ y viceversa. En la esfera de Bloch, esta operación es una rotación de π alrededor del eje x .

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

- **Puerta Z:** Introduce una fase de π cuando el qubit está en $|1\rangle$, pero no altera el $|0\rangle$. Geométricamente, es una rotación de π alrededor del eje z en la esfera de Bloch.

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

- **Puerta Hadamard (H):** Esta puerta es especial, transforma $|0\rangle$ en $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ y $|1\rangle$ en $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. Esto pone el estado en superposición.

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

- **Rotaciones Arbitrarias:** A veces, necesitas ajustar con precisión el estado de un cúbit. Para eso están las puertas $R_x(\theta)$, $R_y(\theta)$ o $R_z(\theta)$, que representan giros en torno a los ejes x , y o z , respectivamente. Estas puertas permiten controlar con precisión cómo se combinan la fase y la amplitud de una superposición.

Puertas Cuánticas de 2 o más Qubits

Con estas puertas es donde el interés por la computación cuántica se vuelve más evidente. Al operar sobre varios qubits al mismo tiempo, estas puertas permiten entrelazarlos, abrir caminos hacia el paralelismo masivo y la computación que desafía la intuición clásica.

- **Puerta CNOT:** Si el qubit de control está en $|1\rangle$, esta puerta aplica una X al qubit objetivo, y si está en $|0\rangle$, no hace nada. Esta puerta es fundamental para crear y manipular estados entrelazados.

$$\text{CNOT} = \begin{cases} |0\rangle |x\rangle \mapsto |0\rangle |x\rangle, \\ |1\rangle |x\rangle \mapsto |1\rangle |\bar{x}\rangle, \end{cases}$$

- **Puerta CZ:** Introduce una fase negativa sólo si ambos qubits están en $|1\rangle$. En todos los demás casos, deja los estados tal y como estaban.

$$\text{CZ} = \begin{cases} |1\rangle |1\rangle \mapsto -|1\rangle |1\rangle, \\ \text{resto inalterado.} \end{cases}$$

- **SWAP:** Como su nombre sugiere, intercambia los estados de dos cúbits. Es una herramienta útil para reorganizar la información cuántica.

$$\text{SWAP}(|a\rangle |b\rangle) = |b\rangle |a\rangle$$

Además de estas puertas, existen otras igualmente fascinantes, como la Toffoli, la puerta de fase controlada o la Fredkin, que permiten construir operaciones más complejas y son esenciales en algunas aplicaciones avanzadas. Sin embargo, para los propósitos de este Trabajo Fin de Máster, las puertas explicadas en este apartado son más que suficientes. Con ellas se puede entender el funcionamiento fundamental de los circuitos cuánticos, explorar el entrelazamiento y, sobre todo, sentar una base sólida para adentrarse en el mundo de la computación cuántica sin perderse en tecnicismos.

Ejemplo de Circuito Cuántico

Supongamos que partimos de dos qubits, ambos en estado $|0\rangle$. Primero aplicamos una puerta Hadamard al primero, y luego una CNOT con ese mismo qubit como control y el segundo como objetivo. Finalmente, medimos ambos.

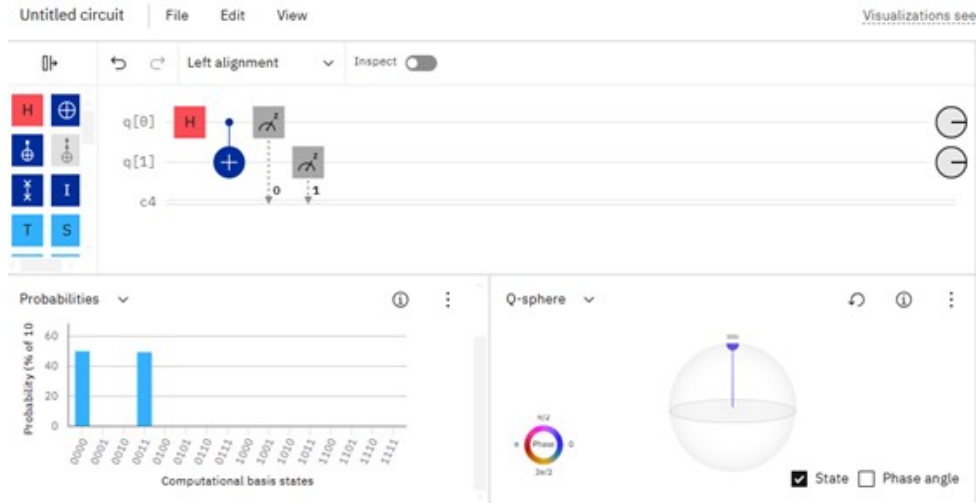


Figura 4: Circuito cuántico estado de Bell [3]

El resultado de este circuito es un estado entrelazado muy especial, conocido como estado de Bell.

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Cuando ejecutamos este circuito en un simulador ideal o un dispositivo cuántico sin ruido, las mediciones arrojan dos resultados posibles, $|00\rangle$ o $|11\rangle$, cada uno con un 50 % de probabilidad. Y lo interesante es que nunca veremos solo uno de los qubits en $|0\rangle$ o $|1\rangle$ por separado, ambos están completamente correlacionados. Es decir, están entrelazados. Lo que le pase a uno afecta al otro, sin importar la distancia.

Y es que eso es lo que hace a la computación cuántica tan interesante, no solo se trata de velocidad o potencia, sino de una manera completamente distinta de procesar la información.

4. Estado del arte

Este capítulo repasa algunos de los avances más destacados en QNLP, comenzando por modelos discriminativos y estructuras fundamentales que han permitido una primera aproximación al lenguaje desde un enfoque cuántico. Más adelante, se analizan con mayor profundidad los modelos generativos, una línea de trabajo emergente, compleja y llena de posibilidades.

4.1. La era NISQ

Antes de comenzar a explicar los trabajos más destacados en el campo de QNLP, es importante entender el momento en el que nos encontramos en cuanto a hardware cuántico real. La era NISQ es el término que describe el estado actual de la tecnología en computación cuántica. Se caracteriza por procesadores cuánticos de escala intermedia, que son intrínsecamente “ruidosos”, es decir, altamente sensibles a perturbaciones externas y propensos a errores [15] [16]. En estas máquinas faltan aún sistemas completos de corrección de errores cuánticos, y su tamaño no es lo suficientemente grande para alcanzar una tolerancia a fallos ni para explotar de forma sostenida la llamada supremacía cuántica [17]. John Preskill acuñó el término NISQ en 2018 precisamente para describir este régimen tecnológico intermedio de los ordenadores cuánticos actuales [17].

En el contexto cuántico, “ruido” no hace referencia a un sonido físico, sino a pequeñas perturbaciones ambientales (fluctuaciones de temperatura, vibraciones, interferencias electromagnéticas, etc.) que alteran el estado de los qubits [16]. Debido a fenómenos como la decoherencia cuántica, un cúbit puede perder su superposición de estados con la mínima interferencia del entorno [16]. Asimismo, las puertas cuánticas no son perfectas, cada puerta de dos cúbits tiene una probabilidad significativa de error, típicamente alrededor de 0,1 % o superior en tecnologías actuales [17]. Estos errores acumulados actúan como ruido lógico dentro del cálculo. En conjunto, la fragilidad de los qubits y la imperfección de las puertas hacen que los circuitos cuánticos profundos pierdan fidelidad rápidamente, de hecho, los procesadores NISQ normalmente no pueden ejecutar de forma confiable más de unos pocos cientos de puertas antes de que el resultado se vuelva esencialmente aleatorio [18]. Como consecuencia directa, las computadoras cuánticas NISQ son lentas, pequeñas y poco precisas, lo que limita severamente su utilidad práctica inmediata [16].

Entre las principales limitaciones impuestas por la era NISQ destacan:

- **Sin corrección de errores a gran escala:** Los dispositivos NISQ no son lo suficientemente sofisticados para implementar esquemas de corrección de errores cuánticos de manera continua [17]. Se necesitarían cientos o miles de cúbits físicos para formar un cúbit lógico libre de errores, recurso del que carecemos hoy. Por ello, cualquier error de operación o decoherencia afecta directamente el cálculo y no puede ser revertido o corregido durante la ejecución [17]. Esto impone una duración máxima a los algoritmos cuánticos, solo aquellos que finalizan antes de que los errores se acumulen en exceso podrán dar resultados significativos.
- **Profundidad de circuito muy limitada:** Dada la tasa de error por operación, los algoritmos cuánticos deben ser muy cortos o poco profundos. Añadir muchas capas de compuertas incrementa exponencialmente la probabilidad de que ocurra al menos un error irreparable en el circuito. En la práctica, incluso con mejoras recientes, un circuito cuántico típico no puede ejecutar de forma fiable más que del orden de cientos de operaciones consecutivas [18]. Esto dificulta la implementación de redes cuánticas profundas o algoritmos que requieran gran número de pasos lógicos secuenciales. Entrenar un modelo cuántico con muchas capas de compuertas es inviable en hardware NISQ debido a la degradación de la señal cuántica. Además, desde la perspectiva algorítmica, muchos circuitos variacionales extensos sufren del problema de las mesetas estériles, donde el gradiente de la función de coste se desvanece exponencialmente al aumentar el tamaño o la profundidad del circuito [15].
- **Escalabilidad restringida:** El número de qubits disponibles en hardware actual limita el tamaño de los problemas que podemos abordar. Con decenas de qubits, solo es factible trabajar con datos

o sistemas de dimensión relativamente pequeña. Por ejemplo, en aplicaciones de procesamiento de lenguaje natural cuántico, esto implica que no se pueden representar directamente oraciones complejas o grandes vocabularios en un registro cuántico sin recurrir a simplificaciones drásticas. De forma similar, en tareas generativas, la resolución o complejidad de los datos que puede modelar un circuito cuántico NISQ está acotada por la cantidad de qubits y conexiones disponibles. Este cuello de botella en escala significa que muchas demostraciones de algoritmos cuánticos actuales se limitan a toy models o instancias de pequeño tamaño, suficientes para probar principios pero no para superar a métodos clásicos en aplicaciones de la vida real [16].

Frente a estas restricciones tecnológicas, la comunidad investigadora ha adoptado estrategias de diseño de algoritmos y modelos específicamente adaptadas a la era NISQ:

- **Algoritmos híbridos cuántico-clásicos:** Buena parte de los avances en algoritmos cuánticos se apoyan en un enfoque híbrido, donde un componente cuántico de baja profundidad se integra dentro de un bucle de optimización clásica.
- **Circuitos cuánticos poco profundos y diseñados a medida:** Dada la necesidad de minimizar los errores, los investigadores procuran diseñar circuitos lo más compactos y cortos posible para cada tarea. Se exploran ansätze específicos que logren el mayor efecto con el menor número de compuertas.
- **Simulación clásica y pruebas en circuitos simulados:** Dado que los ordenadores cuánticos reales son limitados, gran parte del desarrollo de modelos cuánticos ocurre apoyándose en simulaciones clásicas de circuitos cuánticos.

4.2. Quantum Natural Language Processing

4.2.1. DisCoCat

El modelo DisCoCat [19] es un enfoque algebraico que permite traducir la estructura del lenguaje humano a representaciones que pueden ser procesadas por sistemas cuánticos, y se asienta como uno de los pilares en el campo del QNLP. Para lograr esto, descompone las oraciones en elementos matemáticos, usa vectores para representar palabras y tensores para capturar las relaciones gramaticales. Después, estos se transforman en circuitos cuánticos.

Este método no es solo teórico, sino que ha mostrado resultados prometedores en tareas prácticas como la clasificación semántica, el análisis de sentimientos y la identificación de roles temáticos.

Una de sus grandes virtudes es que no solo analiza las palabras por separado, sino que también capta cómo estas interactúan entre sí dentro de una oración, teniendo en cuenta su estructura gramatical, lo que es indispensable en el contexto de QNLP.

El modelo utiliza lo que se conoce como diagramas de cadenas para representar las oraciones de forma categórica.

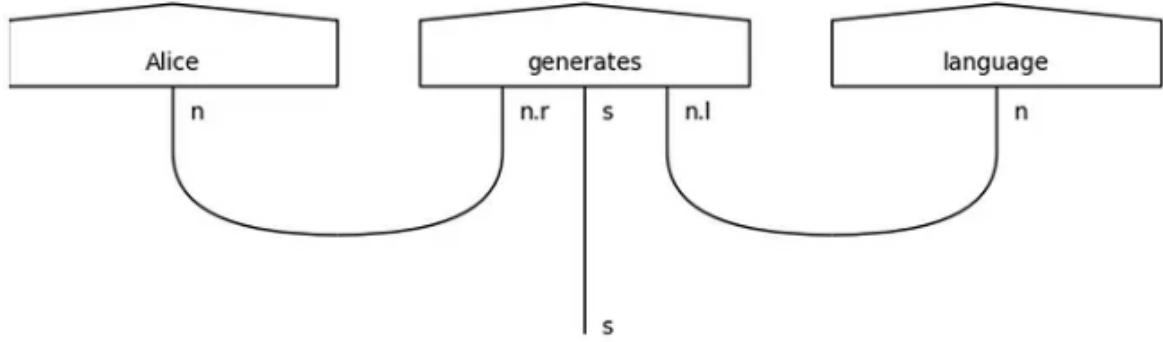


Figura 5: Diagrama de cadenas utilizado como ejemplo en [4]

En estos diagramas, cada palabra aparece como una caja. Los hilos que las conectan indican las relaciones gramaticales, todo basado en las llamadas gramáticas pregrupo. Estos hilos llevan anotaciones que indican su tipo, por ejemplo, n para sustantivos, s para oraciones completas, y variantes como $n.l$ o $n.r$ para señalar si un sustantivo debe ir a la izquierda o a la derecha de otro elemento.

En el ejemplo de la Figura 5, la oración “Alice generates language” se representa con una caja para cada palabra e hilos que conectan a Alice y language con el verbo generates. El resultado es un único hilo abierto (s) que indica que toda la oración es gramaticalmente correcta y tiene sentido como unidad.

Estos diagramas pueden transformarse directamente en circuitos cuánticos parametrizados. Esto es posible gracias a la cercanía formal entre los diagramas categóricos y las operaciones que se usan en computación cuántica. Como se explica en [4], este proceso puede automatizarse mediante Lambeq [20], una biblioteca desarrollada por Cambridge Quantum.

Con Lambeq, el usuario puede ingresar cualquier oración. La herramienta genera automáticamente su correspondiente diagrama de cadenas usando gramáticas pregrupo. Luego, cada palabra se traduce a un estado cuántico inicial y las relaciones gramaticales se convierten en puertas cuánticas que actúan sobre esos estados. El resultado es un circuito cuántico parametrizado que conserva tanto la semántica como la estructura gramatical de la oración [4].

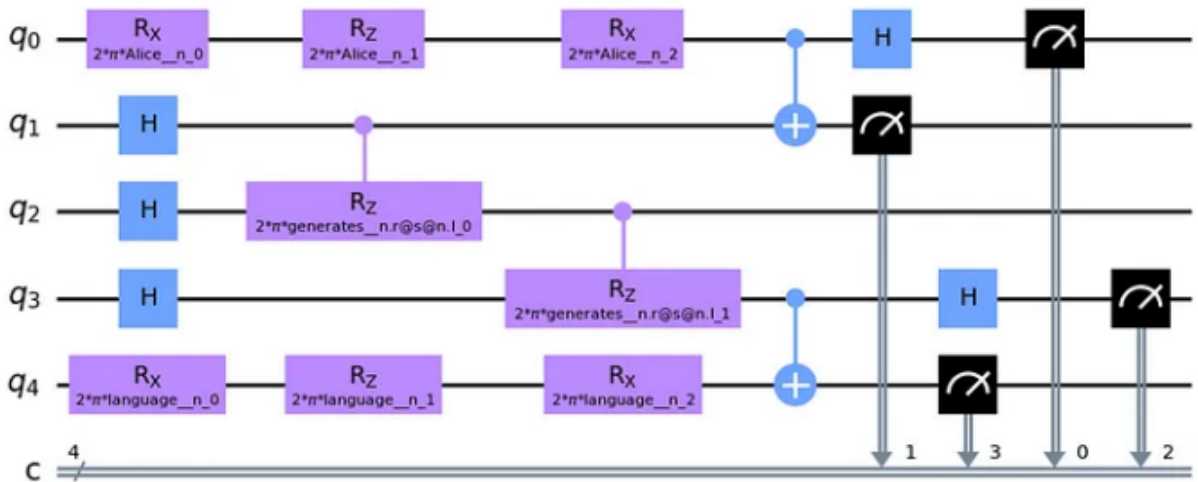


Figura 6: Oración “Alice generates language” convertida a circuito cuántico [4]

En [4] se aplica este enfoque para construir un algoritmo de clasificación siguiendo estos pasos:

1. **Preparación del Corpus:** Primero, se etiqueta un conjunto de oraciones, asignándoles una categoría temática. En este caso, se usó una colección de titulares de noticias, con temas como deportes, política, tecnología y entretenimiento.
2. **Conversión a Circuitos Cuánticos:** Cada oración se transforma en un circuito cuántico parametrizado utilizando lambeq. Si dos oraciones comparten palabras, los circuitos también comparten parámetros, lo que hace el proceso más eficiente.
3. **Entrenamiento del Modelo:** Se ajustan los parámetros de los circuitos para que los bitstrings obtenidos al ejecutarlos coincidan con las categorías asignadas. Por ejemplo, si “Real Madrid wins Champions League” pertenece a la categoría deportes, su resultado cuántico podría ser el bitstring “00”.
4. **Clasificación de Nuevas Oraciones:** Para una oración nueva, se genera su circuito con Lambeq, se ejecuta usando los parámetros optimizados y se interpreta el bitstring final para determinar su categoría.

4.2.2. QSANN

Las Quantum Self-Attention Neural Networks son en esencia una versión cuántica de los mecanismos de autoatención que ya funcionan en transformers como BERT o GPT. Al dar el salto al mundo cuántico, estas redes aprenden las relaciones sutiles que existen entre las palabras, y es que en lenguaje natural, el significado siempre depende del contexto que lo acompaña.

En el estudio [21], QSANN mostró ser más eficiente que su contraparte clásica. En el conjunto MC alcanzó un 100 % de precisión en entrenamiento y pruebas con apenas 25 parámetros, mientras que DisCoCat se quedó en un 79,8 % necesitando 40 parámetros. En el dataset RP, el modelo cuántico volvió a imponerse, aunque con algún tropiezo puntual cuando la tarea exigía una precisión milimétrica.

Además, con la incorporación de mediciones POVM propuestas en [22], se elevó la precisión sin aumentar el número de parámetros. El resultado combina la eficiencia de la computación cuántica para modelar contexto con la flexibilidad de los modelos de autoatención.

Todo esto hace que los QSANN encajen de maravilla en tareas como análisis de sentimientos o clasificación temática, donde descifrar relaciones contextuales es clave. Además, su habilidad para atrapar dependencias no triviales podría convertirse en un salto de calidad para la traducción automática entre idiomas.

Sin embargo, no son todo ventajas, estos modelos siguen lidiando con el ruido propio de los dispositivos cuánticos actuales y con la dificultad de escalar cuando el problema lingüístico se vuelve tan enrevesado como un trabalenguas. Aun así, el horizonte parece prometedor, cada avance acerca un poco más esa mezcla de eficiencia cuántica y sensibilidad lingüística que, hasta hace poco, sonaba a ciencia ficción.

4.2.3. QRNN

Las Quantum Recurrent Neural Networks son en esencia la versión cuántica de las recurrentes clásicas. Mantienen la idea de ir procesando la secuencia paso a paso, pero en un circuito cuántico que aprovecha la superposición para mirar varios hilos temporales a la vez.

En QNLP, esta habilidad es muy valorada. Los textos largos, los contextos históricos o cualquier texto que necesite recordar lo dicho hace varios párrafos se benefician enormemente de una red que no pierde el hilo. El reto es que, sin una quantum RAM que preserve estados intermedios, montar una QRNN completa se vuelve extremadamente complejo. Para sortear el problema, los enfoques más recientes apuestan por

arquitecturas end-to-end que procesan la secuencia de un tirón y evitan la necesidad de almacenar estados intermedios.

Según el trabajo de [23], las QRNN lograron rendimientos comparables e incluso superiores a los de sus contrapartes clásicas en tareas de clasificación de texto, a pesar de las restricciones de hardware, lo que indica que estos enfoques van por buen camino.

No obstante, conviene destacar que hasta que la memoria cuántica no sea una realidad, las QRNN seguirán tropezando cuando la secuencia se alargue demasiado. Aun así, el potencial está ahí, solo falta que la tecnología lo acompañe.

4.3. Modelos generativos cuánticos

Si bien las líneas de investigación exploradas hasta ahora han sido clave para sentar las bases del campo, existe un área que empieza a cobrar especial relevancia y sobre la que está enfocada este Trabajo Fin de Máster, la generación de lenguaje natural cuántico.

En las siguientes secciones se presentan distintos enfoques de generación mediante computación cuántica, cada uno con sus propias fortalezas, limitaciones y niveles de madurez. Aunque no todos los trabajos mencionados son para tareas de generación de lenguaje, es interesante explorarlos y reflexionar sobre cómo podrían usarse esos enfoques para los objetivos del presente Trabajo Fin de Máster.

4.3.1. Q1Synth

Q1Synth, el instrumento presentado en [24], es en pocas palabras, un sintetizador que en lugar de cables y perillas, se alimenta del vaivén cuántico de un qubit. Aprovecha los vectores de estado de un qubit y sus mediciones para crear sonidos. Además, se muestra como una esfera de Bloch donde el intérprete toca “el instrumento” haciendo girar el estado cuántico.

Ahora bien, aunque Q1Synth trabaja en generación de música, sus principios invitan a imaginar otras posibilidades. ¿Y si lleváramos esa misma idea a QNLP? Podríamos modelar palabras o frases como estados cuánticos que codifican significados y matices contextuales, y luego, visualizarlas igual que la esfera para entender cómo interactúan entre sí.

Sin embargo, el salto no es trivial. Las estructuras lingüísticas son mucho más densas y multidimensionales que el paisaje relativamente sencillo que necesitamos para generar un sonido. Para dar ese paso, harían falta avances serios tanto en hardware cuántico como en herramientas de visualización capaces de pintar esas nubes de significado sin perder detalle.

Por ahora, Q1Synth funciona sobre todo como demostración de que la computación cuántica puede convertirse en un instrumento artístico, pero de momento, no es un modelo práctico para procesar lenguaje.

4.3.2. Basado en clasificación

El modelo DisCoCat también puede aplicarse a la generación de texto. En [4] los autores parten de su algoritmo de clasificación (explicado en la sección 4.2.1) y lo transforman en un generador mediante técnicas de optimización combinatoria. Realmente lo que hacen es convertir la búsqueda de una oración adecuada en un problema de “encontrar el mejor movimiento posible”, de modo que la frase resultante sea coherente tanto gramatical como semánticamente dentro de un tema concreto.

El procedimiento avanza de la siguiente forma:

- El usuario indica una temática que ya exista en el corpus.

- Se crea una oración inicial al azar.
- Con la biblioteca Python Lambeq [20], se construye un circuito cuántico parametrizado que codifica la estructura gramatical y el significado de esa frase inicial.
- El circuito se ejecuta varias veces usando los parámetros óptimos extraídos del modelo de clasificación anterior. La salida se vincula a una categoría del tema (por ejemplo, 00 para deportes o 01 para política). De ahí se calcula la probabilidad $P(C)$ de que el circuito devuelva la etiqueta correcta.
- Cuando $P(C)$ cae por debajo de un umbral, en este caso 90%, se generan variantes de la frase C' cambiando, insertando o quitando palabras. Se vuelve a evaluar y, si $P(C') > P(C)$, la nueva oración toma el relevo.
- El ciclo se repite hasta que la frase alcanza el nivel de coherencia buscado.

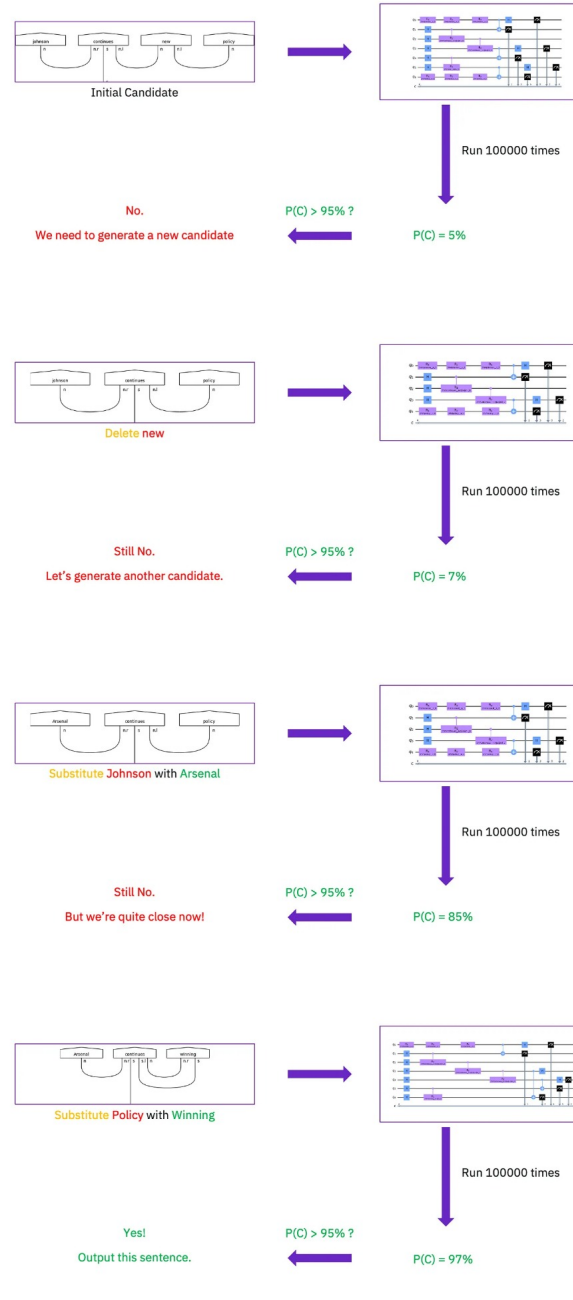


Figura 7: Representación gráfica del algoritmo descrito [4].

Por ahora, el algoritmo no supera a los métodos clásicos en rapidez ni en precisión. Aun así, demuestra que los sistemas cuánticos pueden abordar desafíos lingüísticos y deja la puerta abierta a nuevas investigaciones.

4.3.3. QGANs

El artículo [25] presenta un enfoque híbrido cuántico-clásico para cargar distribuciones de probabilidad en estados cuánticos de manera eficiente. Se introduce el uso de Quantum Generative Adversarial Networks

para facilitar el aprendizaje y la carga aproximada de distribuciones de probabilidad a partir de datos clásicos.

Cargar datos en estados cuánticos con total fidelidad exige $\mathcal{O}(2^n)$ compuertas para n qubits, una cantidad demasiado alta para el hardware disponible a día de hoy. La solución propuesta en [25] consiste en renunciar a la perfección milimétrica y aceptar una aproximación entrenada. Las qGANs reducen el esfuerzo a un coste polinómico $\mathcal{O}(\text{poly}(n))$, suficiente para que algoritmos como Quantum Amplitude Estimation puedan desplegar su potencial.

Las GANs clásicas emplean dos redes neuronales, un generador y un discriminador. El generador produce datos, mientras que el discriminador trata de distinguir entre datos reales y generados. Las QGANs mencionadas en [25], reemplazan el generador por un generador cuántico y mantienen el discriminador como una red neuronal clásica.

El funcionamiento de estas qGANs es el siguiente:

- **Generador cuántico:** Se implementa con circuitos variacionales compuestos por rotaciones R_Y y compuertas controladas CZ . El generador toma un estado de entrada $|\psi_{\text{in}}\rangle$ y lo transforma en un estado de salida $|g_\theta\rangle$ que representa la distribución deseada.
- **Discriminador Clásico:** Una red neuronal que clasifica las muestras como reales o generadas.
- **Entrenamiento:** Optimización de funciones de pérdida para el generador y el discriminador de forma alternada utilizando el optimizador AMSGRAD.

En [25], se realizaron experimentos de simulación cuántica con tres distribuciones de probabilidad, log-normal, triangular y bimodal. Durante estos experimentos, se exploraron distintas inicializaciones (uniforme, normal y aleatoria) para el estado de entrada del generador cuántico. Los resultados demostraron que el rendimiento del modelo mejora al aumentar la profundidad del circuito y al seleccionar una inicialización adecuada para el estado de entrada.

Además, se aplicó este enfoque en el campo de finanzas cuánticas, entrenaron una qGAN para representar la distribución log-normal del precio de un activo subyacente y, con ayuda de QAE, estimaron el valor justo de una opción europea. El resultado fue una mejora cuadrática en precisión frente a las Monte Carlo clásicas.

Por último, trasladaron el experimento al procesador IBM Q Boeblingen de 20 qubits. A pesar del ruido, se consiguió un generador capaz de aproximar la distribución deseada.

Al igual que en aplicaciones financieras y de simulación, las capacidades únicas de las qGANs podrían extenderse al modelado de estructuras lingüísticas complejas:

- **Dependencias contextuales.** Un generador cuántico podría absorber patrones de co-ocurrencia de palabras en distintos ambientes, mientras el discriminador vigila la coherencia semántica del lenguaje producido.
- **Compresión de corpus masivos.** Al codificar patrones lingüísticos como estados cuánticos compactos, es posible imaginar guardar bibliotecas enteras en unos cuantos bloques de qubits.

Para llevar estas hipótesis a la realidad necesitamos qubits más estables y herramientas de visualización capaces de desentrañar nubes de significado sin desdibujar matices. Aun así, las QqGANs ya han demostrado que pueden manejar distribuciones complejas, el salto al lenguaje es cuestión de tiempo.

4.3.4. QCBMs

Una idea bastante interesante que ha surgido en los últimos años son las Quantum Circuit Born Machines, como se presenta en [26]. Se trata de una forma novedosa de modelar distribuciones de probabilidad usando circuitos cuánticos parametrizados.

La idea detrás de las QCBMs es aprovechar la potencia del mundo cuántico para trabajar con espacios de alta dimensionalidad, y así poder aprender cómo se distribuyen ciertos datos. Su objetivo principal es aproximarse lo más posible a una distribución de probabilidad objetivo, $q(x)$, generando muestras de otra distribución $p_\theta(x)$ que se ajusta poco a poco al optimizar los parámetros cuánticos.

Cada QCBM está formada por tres piezas clave, un circuito cuántico parametrizado, una función de coste y un optimizador clásico.

Primero, el circuito comienza con el sistema cuántico inicializado en el estado base $|0\rangle^{\otimes n}$. Luego, ese estado pasa por una serie de puertas cuánticas ajustables, representadas como $U(\theta)$, que van modificando el estado según los valores de sus parámetros. Cuando medimos este sistema, la probabilidad de obtener un resultado específico x viene dada por la Regla de Born:

$$p_\theta(x) = |\langle x|U(\theta)|0\rangle|^2$$

Esta fórmula es increíblemente útil, nos permite codificar distribuciones complejas de manera eficiente, algo que sería muy difícil con métodos clásicos.

Luego entra en juego la función de coste, cuya labor es comparar cómo de cerca está la distribución aprendida $p_\theta(x)$ de la verdadera $q(x)$. Para hacerlo, se usan métricas especializadas como la divergencia de Kullback-Leibler [27] o la Discrepancia Máxima de Medias [28]. Estas herramientas miden la diferencia entre ambas distribuciones y nos dan una pista de hacia dónde tenemos que mover los parámetros θ .

Finalmente, todo esto se conecta con un optimizador clásico. Aunque el modelo sea cuántico, la parte del ajuste sigue dependiendo de algoritmos clásicos de optimización. El objetivo es minimizar esa función de coste para que la máquina vaya mejorando poco a poco.

Según [26], estas máquinas ya han sido probadas experimentalmente, por ejemplo en [29], donde se exploró su potencial. Una de las aplicaciones fue generar imágenes sintéticas del conjunto “Bars and Stripes”. Este conjunto contiene matrices binarias pequeñas, de 2×2 o 3×3 píxeles, que representan patrones simples de barras horizontales o verticales. Puede parecer básico, pero es un buen punto de partida para probar modelos generativos.

Lo curioso es que las QCBMs lograron aprender estas distribuciones con bastante precisión, lo cual ya es un paso importante. Mostraron que pueden capturar patrones discretos con éxito, algo que puede ser clave en muchos contextos.

Otro experimento interesante realizado en [29] fue usar las QCBMs para preparar estados térmicos cuánticos. En términos sencillos, se trataba de simular distribuciones que representaran sistemas físicos en equilibrio térmico. Aunque los resultados fueron prometedores, también mostraron un problema recurrente, los circuitos profundos aún enfrentan limitaciones por el ruido del hardware actual. Pero eso no quita mérito al enfoque, más bien, señala por dónde hay que seguir trabajando.

Estos estudios confirman que las QCBMs tienen potencial real como modelos generativos, y quizá sean la base para aplicaciones futuras en campos como el procesamiento del lenguaje natural cuántico.

Si las relaciones semánticas y gramaticales pudieran representarse como distribuciones en espacios de Hilbert, entonces podríamos usar las QCBMs para modelar cómo las palabras se relacionan entre sí, cómo cambia el significado según el contexto o incluso cómo surgen ambigüedades.

Además, su habilidad para operar en espacios de alta dimensionalidad podría permitirnos construir modelos capaces de manejar grandes cantidades de texto con menos recursos computacionales que los métodos

clásicos actuales. Esto no solo sería más eficiente, sino que abriría nuevas formas de entender el lenguaje desde una perspectiva cuántica.

Y es que uno de los retos del procesamiento del lenguaje es justamente eso, capturar esas sutilezas que a veces ni siquiera notamos cuando hablamos. Las QCBMs, gracias a su capacidad para modelar distribuciones complejas, podrían ayudarnos a detectar y reproducir esos matices, como las ambigüedades contextuales, que tanto cuestan a los modelos tradicionales.

Aunque todavía estamos en los primeros pasos, las Quantum Circuit Born Machines parecen tener mucho que ofrecer. No solo por su elegancia teórica, sino por su potencial práctico, y si logramos superar algunos de los desafíos técnicos actuales, quizás estemos ante una herramienta clave para el futuro de la inteligencia artificial cuántica.

4.3.5. QBMs

En [26] también se mencionan las Quantum Boltzmann Machines , o QBMs, que fueron introducidas en el trabajo de [30]. Estas máquinas vienen a ser como la versión cuántica evolucionada de las clásicas Boltzmann Machines.

La diferencia fundamental está en que, mientras las originales usaban unidades clásicas y energías definidas por conexiones simples, las QBMs reemplazan todo eso, los nodos pasan a ser qubits, y las energías se describen ahora mediante un Hamiltoniano cuántico. Y es que esto último abre una puerta a representar relaciones mucho más complejas entre los datos.

Gracias a este cambio, las QBMs pueden capturar interacciones mucho más complejas que las versiones clásicas. Hay distribuciones de probabilidad que simplemente son inalcanzables para una máquina de Boltzmann tradicional, pero que con la ayuda del mundo cuántico empiezan a hacerse visibles.

Una de las aplicaciones más interesantes que han tenido las QBMs es en el aprendizaje generativo supervisado y en la simulación de distribuciones de Bernoulli mezcladas. Esto significa que han sido capaces de aprender patrones probabilísticos complejos a partir de datos simples, y hacerlo mejor que sus contrapartes clásicas cuando el problema se complica.

No obstante, el entrenamiento de estas máquinas también tiene sus desventajas. Por ejemplo, calcular las fases positiva y negativa, dos partes clave del proceso de ajuste, puede volverse difícil cuando el problema crece en tamaño, además del del ruido cuántico actual.

Aunque su uso se ha centrado principalmente en campos como el aprendizaje reforzado o la simulación de estados térmicos cuánticos, su capacidad para modelar distribuciones complejas las hace especialmente interesantes para otras áreas. Una de ellas podría ser el procesamiento del lenguaje natural cuántico, si logramos mapear las relaciones semánticas y gramaticales entre palabras como interacciones entre qubits, entonces se podría usar un Hamiltoniano diseñado a medida para codificar esas estructuras. Palabras relacionadas, frases ambiguas, dependencias sintácticas... todo eso podría verse reflejado en cómo interactúan los distintos qubits dentro del modelo.

Por supuesto, aún estamos lejos de ver modelos cuánticos funcionando en nuestros asistentes de voz o motores de búsqueda, pero estos primeros pasos sugieren que, con tiempo y mejoras en hardware, quizá podamos empezar a pensar en el lenguaje desde una perspectiva distinta.

5. Desarrollo del proyecto

5.1. Quantum Generative Adversarial Network

Como primer paso práctico en mi exploración del Procesamiento Cuántico del Lenguaje Natural, y teniendo en cuenta que era mi primera vez diseñando modelos cuánticos, decidí empezar con algo sencillo, un clasificador aplicado a un subconjunto del dataset BBC News [31]. La idea tenía dos motivaciones claras, por un lado, quería familiarizarme con herramientas clave como Lambeq y PennyLane, y por otro, construir una base sólida para lo que más adelante podría convertirse en un discriminador cuántico dentro de una Quantum Generative Adversarial Network.

5.1.1. Clasificador cuántico

Para comenzar con el clasificador, cargue el conjunto de datos BBC News, que contiene titulares organizados en categorías como deportes, tecnología, entretenimiento, política o negocios. Para este experimento, me enfoqué únicamente en el campo 'title', que contiene el titular de cada noticia.

category	
sport	511
business	510
politics	417
tech	401
entertainment	386

Figura 8: Número de titulares por categoría

El proceso comenzó con un modelo clásico. Entrené un clasificador basado en regresión logística, usando una representación TF-IDF de los textos. Los resultados rondaron el 88 % de precisión, lo cual me dio una referencia sólida para evaluar después el desempeño del modelo cuántico.

	precision	recall	f1-score	support
business	0.77	0.90	0.83	102
entertainment	0.92	0.84	0.88	77
politics	0.93	0.74	0.82	84
sport	0.84	0.93	0.88	102
tech	0.93	0.88	0.90	80
accuracy			0.86	445
macro avg	0.88	0.86	0.86	445
weighted avg	0.87	0.86	0.86	445

Figura 9: Resultados clasificador clásico

El desarrollo del clasificador cuántico fue bastante más complejo, tanto en términos conceptuales como técnicos. Se utilizó la biblioteca Lambeq [20], una herramienta diseñada para implementar procesamiento del lenguaje natural cuántico, basada en la teoría categórica de gramáticas pregrupo y circuitos parametrizados. A diferencia de modelos clásicos, Lambeq requiere que las frases estén en una forma gramaticalmente

sencilla y estructurada, para que el parser pueda analizar correctamente su sintaxis. Esto implicó una serie de pasos:

- Primero fue necesario filtrar los titulares con más de 10 palabras
- También fue necesario eliminar contracciones y signos de puntuación.
- Se añadió el punto final en cada titular para evitar errores gramaticales.
- Por último, se realizó una conversión a minúsculas y eliminación de tokens anómalos.

Una vez limpias, las frases fueron procesadas utilizando BobcatParser, una herramienta que convierte cada oración en un diagrama categórico. Este diagrama representa la estructura gramatical de la frase a través de tipos lingüísticos como n (nombre), s (oración), np (frase nominal), entre otros.

Sin embargo, durante esta etapa aparecieron numerosos errores de *parsing*, principalmente generados por frases con estructuras gramaticales poco comunes, ambigüedad semántica y tokens fuera del vocabulario gramatical del parser. A pesar de aplicar varias estrategias de limpieza, fue necesario descartar una parte significativa del dataset debido a estos errores, lo que evidencia una diferencia clave respecto al modelo clásico.

Finalmente, se lograron crear 1589 diagramas a partir del dataset original.

Diagrama 1 (Categoría: business)

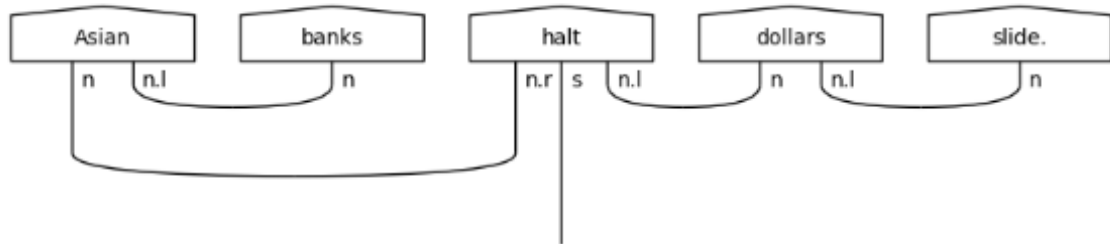


Diagrama 2 (Categoría: sport)

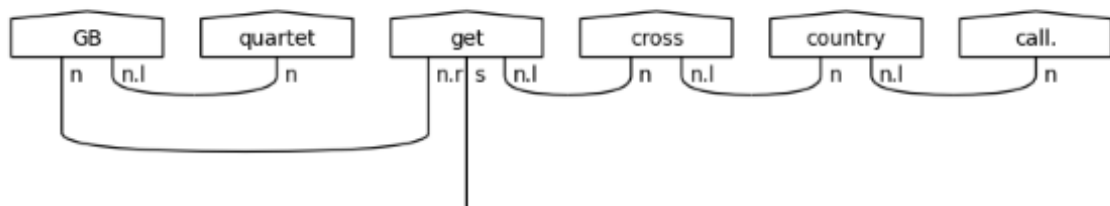


Diagrama 3 (Categoría: entertainment)

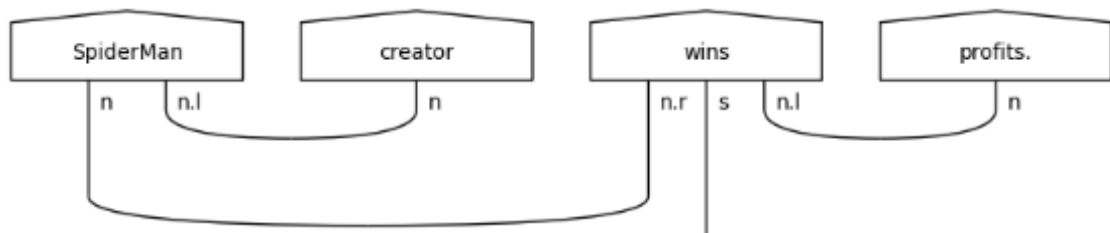


Figura 10: Ejemplos de los diagramas creados con Lambeq

Una vez generados los diagramas gramaticales, el siguiente paso fue convertirlos en circuitos cuánticos parametrizados. Para ello, Lambeq proporciona distintos ansatz, que permiten mapear estructuras

gramaticales en configuraciones específicas de puertas cuánticas. En este trabajo opté por el StronglyEntanglingAnsatz, que se caracteriza por una topología densa en entrelazamientos y es especialmente útil para tareas de clasificación. Antes de realizar esta conversión, se analizaron los tipos atómicos gramaticales presentes en los diagramas generados. Esta información fue utilizada para construir un diccionario de mapeo que asigna un número de qubits a cada tipo gramatical relevante.

Tipos atómicos encontrados:

```
p.l
s.l
p
n
n.r
n.l.l
s
s.r
n.r.r
n.l
```

Figura 11: Tipos atómicos encontrados

Al realizar la conversión de las oraciones a circuitos cuánticos, surgieron errores de RAM debido a la complejidad de los circuitos generados en términos de profundidad y número de qubits. Para poder continuar con el experimento con el hardware disponible, fue necesario volver a reducir el problema, y decidí continuar únicamente con las oraciones de 2 categorías: sport y tech.

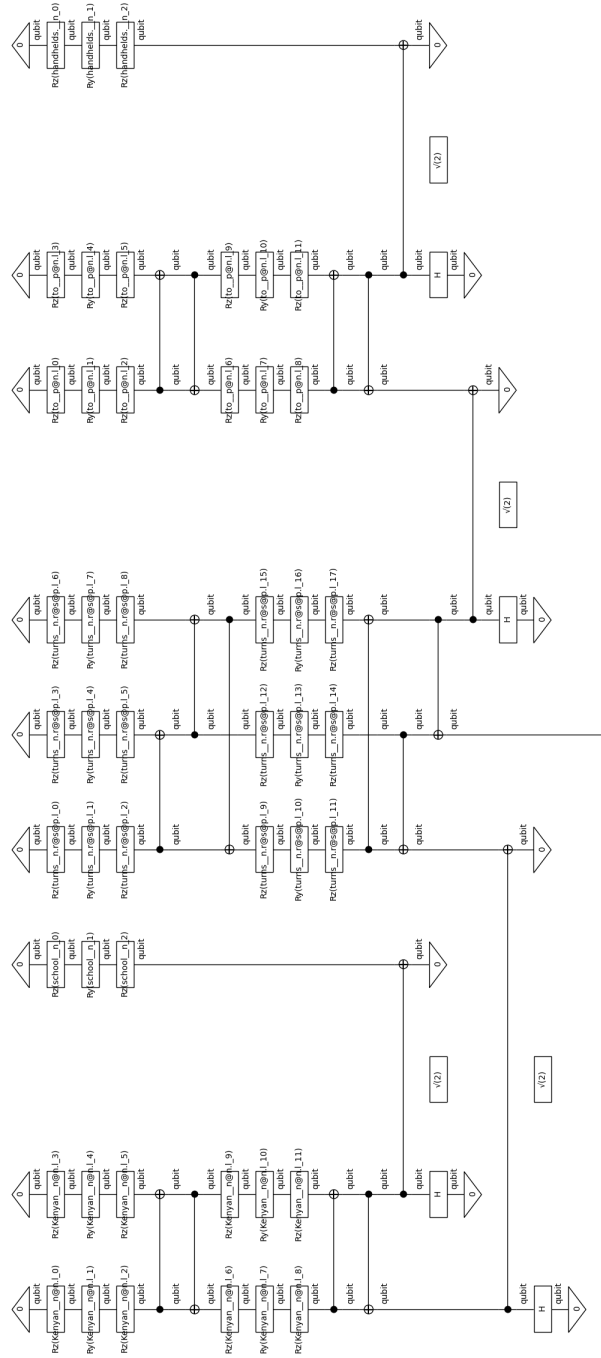


Figura 12: Circuito cuántico generado por Lambeq para una de las oraciones del dataset.

Cada uno de los titulares se convierte en un circuito cuántico como el de la Figura 12, que representa una transformación específica sobre los qubits asignados a la estructura lingüística de la frase. Estos circuitos se utilizan como entrada para el modelo de clasificación cuántica implementado con PennyLane.

Para el modelo cuántico, se empleó el módulo PennyLaneModel de Lambeq, que permite definir modelos cuánticos a partir de diagramas categóricos. Primero se convirtieron las etiquetas sport y tech en valores numéricos 0 y 1 y se preparó un dataset con los circuitos y sus correspondientes clases. El modelo se

configuró con el backend ‘default.qubit’, un simulador cuántico basado en qubits.

El modelo fue entrenado durante 15 épocas con una tasa de aprendizaje de 0.1, utilizando el optimizador Adam y la pérdida cross_entropy. El entrenamiento se llevó a cabo en CPU, y cada época tardó aproximadamente 5 minutos, con un tiempo total de entrenamiento de 1 hora y 18 minutos. La función de evaluación fue la precisión sobre el conjunto de entrenamiento, dado que no se realizó una partición adicional para validación.

```
Epoch 1:  train/loss: 0.4971  valid/loss: ----- train/time: 4m59s  valid/time: ----- train/acc: 0.5291  valid/acc: -----
Epoch 2:  train/loss: 0.7715  valid/loss: ----- train/time: 4m57s  valid/time: ----- train/acc: 0.5963  valid/acc: -----
Epoch 3:  train/loss: 0.4878  valid/loss: ----- train/time: 5m7s   valid/time: ----- train/acc: 0.6376  valid/acc: -----
Epoch 4:  train/loss: 0.4697  valid/loss: ----- train/time: 5m6s   valid/time: ----- train/acc: 0.6667  valid/acc: -----
Epoch 5:  train/loss: 0.4553  valid/loss: ----- train/time: 5m12s  valid/time: ----- train/acc: 0.7401  valid/acc: -----
Epoch 6:  train/loss: 0.6522  valid/loss: ----- train/time: 5m24s  valid/time: ----- train/acc: 0.7309  valid/acc: -----
Epoch 7:  train/loss: 0.5130  valid/loss: ----- train/time: 5m18s  valid/time: ----- train/acc: 0.7309  valid/acc: -----
Epoch 8:  train/loss: 0.3934  valid/loss: ----- train/time: 5m11s  valid/time: ----- train/acc: 0.7492  valid/acc: -----
Epoch 9:  train/loss: 0.4879  valid/loss: ----- train/time: 5m17s  valid/time: ----- train/acc: 0.7446  valid/acc: -----
Epoch 10: train/loss: 0.9114  valid/loss: ----- train/time: 5m18s  valid/time: ----- train/acc: 0.7936  valid/acc: -----
Epoch 11: train/loss: 0.4576  valid/loss: ----- train/time: 5m16s  valid/time: ----- train/acc: 0.7982  valid/acc: -----
Epoch 12: train/loss: 0.6739  valid/loss: ----- train/time: 5m15s  valid/time: ----- train/acc: 0.7951  valid/acc: -----
Epoch 13: train/loss: 0.4840  valid/loss: ----- train/time: 5m19s  valid/time: ----- train/acc: 0.8073  valid/acc: -----
Epoch 14: train/loss: 0.4228  valid/loss: ----- train/time: 5m18s  valid/time: ----- train/acc: 0.8043  valid/acc: -----
Epoch 15: train/loss: 0.6835  valid/loss: ----- train/time: 5m16s  valid/time: ----- train/acc: 0.8089  valid/acc: -----

Training completed!
train/time: 1h18m10s  train/time_per_epoch: 5m13s  train/time_per_step: 4.74s  valid/time: None  valid/time_per_eval: None
Precisión final en el conjunto de entrenamiento: 0.8287
```

Figura 13: Entrenamiento del clasificador.

Al finalizar el entrenamiento, el modelo alcanzó una precisión del 82.87% sobre el conjunto de entrenamiento. Para obtener una evaluación más detallada, se calcularon las métricas de precisión, recall, F1-score y la matriz de confusión.

	precision	recall	f1-score	support
sport	0.84	0.87	0.86	382
tech	0.81	0.78	0.79	272
accuracy			0.83	654
macro avg	0.82	0.82	0.82	654
weighted avg	0.83	0.83	0.83	654

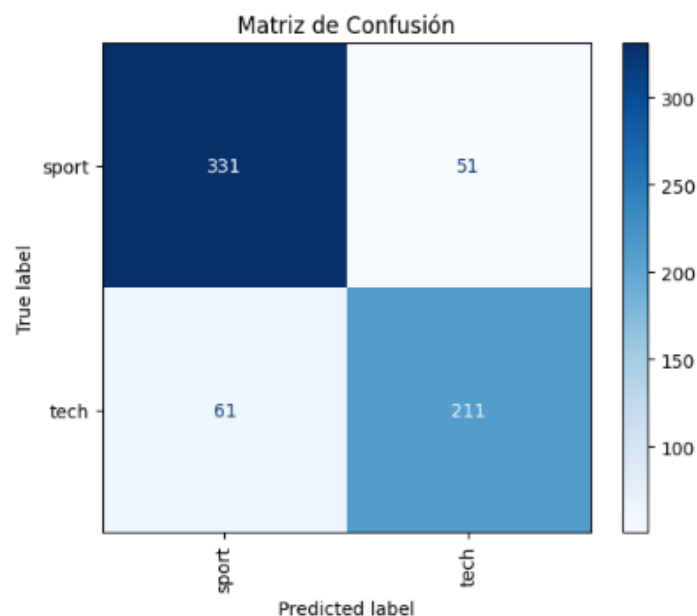


Figura 14: Enter Caption

Los resultados indican un rendimiento equilibrado entre ambas clases. Aunque ligeramente por debajo del modelo clásico, que alcanzó un 88 % de precisión, el modelo cuántico demuestra una capacidad significativa para generalizar a partir de circuitos generados directamente desde la estructura gramatical de las frases.

5.1.2. Discriminador cuántico

El siguiente paso fue diseñar un discriminador cuántico para la Quantum GAN, inspirado en el clasificador desarrollado en el apartado anterior. A diferencia del clasificador que trabajaba con etiquetas categóricas explícitas, el discriminador debía aprender a diferenciar entre frases reales del dataset y frases generadas artificialmente.

Para las frases reales, se seleccionaron oraciones cortas del mismo dataset, aplicando los mismos filtros y técnicas de limpieza que con el clasificador. Para las frases falsas, se generaron variaciones sintéticas a partir de las reales reordenando aleatoriamente las palabras, de esta forma, se mantiene el vocabulario pero se altera la estructura.

Tanto las frases reales como las falsas fueron procesadas con Lambeq utilizando el mismo ansatz cuántico y el mismo mapeo de tipos gramaticales que con el clasificador. Cada oración se convirtió en un circuito cuántico individual, pero durante el proceso, se volvieron a presentar errores de *parsing* en varias frases falsas, lo cual obligó a realizar un muestreo más cuidadoso para equilibrar el conjunto final. Finalmente, se logró generar 27 circuitos de oraciones falsas, que servirían para entrenar al discriminador junto a 33 circuitos de oraciones reales.

```
Error al analizar el fake: talks Juventus hold to with Mutu. Se omitirá. Error: Bobcat failed to parse 'talks Juventus hold to with Mutu'.
Agregado correctamente: to Ireland miss match Wilkinson
Agregado correctamente: truce in battle deficit EC calls
Error al analizar el fake: Newcastle up Babayaro line. Se omitirá. Error: Bobcat failed to parse 'Newcastle up Babayaro line'.
Error al analizar el fake: Fry role for Hitchhikers in set. Se omitirá. Error: Bobcat failed to parse 'Fry role for Hitchhikers in set'.
Agregado correctamente: tsunami join Top show TV US stars
Agregado correctamente: Mido third makes apology
Agregado correctamente: shares Asia defy gloom postquake
Agregado correctamente: UK for pitches ethnic Howard vote
Agregado correctamente: blogging the bug gets Microsoft
Generados 27 diagramas falsos válidos.
Número de circuitos reales: 33
Número de circuitos falsos: 27
```

Figura 15: Ejemplo de frases aceptas y rechazadas por BobcatParser

El discriminador fue entrenado usando el módulo PennyLaneModel con una salida binaria. Dado que el tamaño del dataset era reducido, se utilizó un batch size mínimo y se entrenó el modelo durante 15 épocas. A lo largo del entrenamiento, el modelo fue capaz de aprender de manera progresiva, alcanzando una precisión final del 94.4 % sobre el conjunto de entrenamiento.

```
Epoch 1: train/loss: 0.8873 valid/loss: ----- train/time: 1m13s valid/time: ----- train/acc: 0.5185 valid/acc: -----
Epoch 2: train/loss: 0.4163 valid/loss: ----- train/time: 1m14s valid/time: ----- train/acc: 0.5556 valid/acc: -----
Epoch 3: train/loss: 0.8659 valid/loss: ----- train/time: 1m14s valid/time: ----- train/acc: 0.6296 valid/acc: -----
Epoch 4: train/loss: 0.3203 valid/loss: ----- train/time: 1m12s valid/time: ----- train/acc: 0.7778 valid/acc: -----
Epoch 5: train/loss: 0.8263 valid/loss: ----- train/time: 1m12s valid/time: ----- train/acc: 0.6667 valid/acc: -----
Epoch 6: train/loss: 0.4554 valid/loss: ----- train/time: 1m52s valid/time: ----- train/acc: 0.7778 valid/acc: -----
Epoch 7: train/loss: 0.3894 valid/loss: ----- train/time: 1m22s valid/time: ----- train/acc: 0.8333 valid/acc: -----
Epoch 8: train/loss: 0.4590 valid/loss: ----- train/time: 1m14s valid/time: ----- train/acc: 0.8704 valid/acc: -----
Epoch 9: train/loss: 0.8001 valid/loss: ----- train/time: 1m15s valid/time: ----- train/acc: 0.8519 valid/acc: -----
Epoch 10: train/loss: 0.4838 valid/loss: ----- train/time: 1m14s valid/time: ----- train/acc: 0.8333 valid/acc: -----
Epoch 11: train/loss: 0.3224 valid/loss: ----- train/time: 1m14s valid/time: ----- train/acc: 0.8704 valid/acc: -----
Epoch 12: train/loss: 0.3457 valid/loss: ----- train/time: 1m17s valid/time: ----- train/acc: 0.9444 valid/acc: -----
Epoch 13: train/loss: 0.3298 valid/loss: ----- train/time: 1m14s valid/time: ----- train/acc: 0.9259 valid/acc: -----
Epoch 14: train/loss: 1.1118 valid/loss: ----- train/time: 2m13s valid/time: ----- train/acc: 0.9444 valid/acc: -----
Epoch 15: train/loss: 0.3178 valid/loss: ----- train/time: 1m28s valid/time: ----- train/acc: 0.9630 valid/acc: -----

Training completed!
train/time: 20m27s train/time_per_epoch: 1m22s train/time_per_step: 1.52s valid/time: None valid/time_per_eval: None
Precisión final en el conjunto de entrenamiento: 0.9444
```

Figura 16: Entrenamiento del discriminador.

Estos resultados muestran que incluso con un conjunto limitado de muestras, el modelo es capaz de

distinguir patrones estructurales que separan las frases auténticas de las construidas aleatoriamente, lo que le convierte en un buen candidato para la QGAN.

5.1.3. Generador y Quantum GAN

Después de entrenar el discriminador cuántico, llegó el momento de construir una Quantum GAN completa. La idea era a priori sencilla, combinar un generador clásico con nuestro discriminador cuántico con el objetivo de generar frases que fuesen capaces de engañar al discriminador, es decir, que parecieran auténticas.

Dado que herramientas como Lambeq aún no ofrecen generadores cuánticos completamente desarrollados, se optó por usar una red LSTM clásica como generador, por lo que el sistema resultante terminó siendo híbrido.

La lógica del entrenamiento se basa en un enfoque simplificado de policy gradient. En cada paso, el generador produce una secuencia textual. Esta secuencia se transforma en texto y es evaluada por el discriminador cuántico. El valor de recompensa es la probabilidad de que el discriminador clasifique la frase como real. La pérdida del generador se calcula mediante la fórmula:

$$\text{loss} = -\log(p_{\text{generador}}) \cdot \text{reward}$$

Gracias a esta fórmula, cuanto más falsa sea la frase, mayor será la penalización.

Para evitar saturar la memoria, se limitó el vocabulario a las 500 palabras más comunes y se usaron solo titulares de hasta 8 palabras. El generador fue inicializado desde cero y entrenado durante 10 épocas, y el discriminador se cargó de la ejecución anterior.

```

Epoch 1/10
  gen_text: 'open for tottenham defy spiderman plots'  reward=0.0000  loss=0.0000

Epoch 2/10
(Info) Diagrama no visto: 'betting school wins philippoussis rusedski makes returns boss'. Se asigna prob=0.0
  gen_text: 'betting school wins philippoussis rusedski makes returns boss'  reward=0.0000  loss=0.0000

Epoch 3/10
(Info) Diagrama no visto: 'gloom worlds spotlight edwards benitez sundance running call'. Se asigna prob=0.0
  gen_text: 'gloom worlds spotlight edwards benitez sundance running call'  reward=0.0000  loss=0.0000

Epoch 4/10
(Info) Diagrama no visto: 'goes juventus gloom profits warning miss payoff police'. Se asigna prob=0.0
  gen_text: 'goes juventus gloom profits warning miss payoff police'  reward=0.0000  loss=0.0000

Epoch 5/10
(Info) Diagrama no visto: 'pipeline 1m more new wants bnp tips bmw'. Se asigna prob=0.0
  gen_text: 'pipeline 1m more new wants bnp tips bmw'  reward=0.0000  loss=0.0000

Epoch 6/10
(Info) Diagrama no visto: 'microsoft stop 1638m state faces media us police'. Se asigna prob=0.0
  gen_text: 'microsoft stop 1638m state faces media us police'  reward=0.0000  loss=0.0000

Epoch 7/10
(Info) Diagrama no visto: 'rescued lengthy bug high state card third mosque'. Se asigna prob=0.0
  gen_text: 'rescued lengthy bug high state card third mosque'  reward=0.0000  loss=0.0000

Epoch 8/10
(Info) Diagrama no visto: 'semifinals explosion tips police high uk 1638m saudi'. Se asigna prob=0.0
  gen_text: 'semifinals explosion tips police high uk 1638m saudi'  reward=0.0000  loss=0.0000

Epoch 9/10
(Info) Diagrama no visto: 'duo box to school worlds the fry dialler'. Se asigna prob=0.0
  gen_text: 'duo box to school worlds the fry dialler'  reward=0.0000  loss=0.0000

Epoch 10/10
(Info) Diagrama no visto: 'id lloyds microsoft ec bug jobs newcastle charge'. Se asigna prob=0.0
  gen_text: 'id lloyds microsoft ec bug jobs newcastle charge'  reward=0.0000  loss=0.0000

```

Figura 17: Entrenamiento de la QGAN

Aunque la implementación funcionó correctamente desde el punto de vista técnico, surgió un obstáculo crítico. Resulta que el discriminador cuántico, por cómo funciona internamente, no puede evaluar frases que no haya visto antes. No es que no las clasifique bien, simplemente no puede procesarlas. Esto se debe a que internamente, los diagramas categóricos son almacenados en un mapa, y si una oración generada no fue procesada previamente y convertida a circuito, el sistema lanza una excepción.

Este comportamiento se traduce en que todas las frases generadas recibían una recompensa de cero, lo que hacía que la pérdida también fuera cero, y si no hay pérdida, no hay aprendizaje.

Este resultado, aunque no era lo esperado, resulta útil para futuras investigaciones, muestra que, si se desea usar QGANs con discriminadores categóricos, es necesario un mecanismo de parsing online o precompilación masiva de posibles frases, algo que no parece factible con las herramientas disponibles.

5.2. Generación de lenguaje mediante Quantum Circuit Born Machines

Las Quantum Circuit Born Machines son un modelo generativo cuántico empleado para aprender distribuciones probabilísticas. La idea de emplear este tipo de circuitos para la generación de lenguaje es aprovechar su capacidad para estimar distribuciones probabilísticas que podrían representar la estructura estadística del lenguaje natural.

Es importante señalar que los experimentos de este apartado han sido ejecutados en un simulador de computación cuántica utilizando PennyLane en modo ideal, sin ruido ni muestreo finito. Esto posibilita

obtener probabilidades precisas y simplifica la comprobación y evaluación del método de entrenamiento, aunque puede variar de la implementación en hardware cuántico real.

5.2.1. Generar caracteres con 2 qubits

Este primer experimento se basa en un circuito cuántico analítico que emplea dos qubits, lo que permite la representación de 2^2 estados diferentes. Cada uno de estos estados está vinculado a un símbolo de un pequeño alfabeto.

Los símbolos elegidos fueron: “A”, “B”, “C” y “_”. La codificación se lleva a cabo a través de bitstrings de dos bits, de manera que el estado “00” corresponde al símbolo “A”, “01” a “B”, “10” a “C” y “11” a “_”.

Además, se establece una distribución objetivo de estos 4 estados, la cual representa la probabilidad deseada para cada símbolo. En este caso, la distribución objetivo es: 40 % para “A”, 30 % para “B”, 20 % para “C” y 10 % para “_”.

Símbolo	Bitstring	Probabilidad
A	00	40 %
B	01	30 %
C	10	20 %
_	11	10 %

Tabla 1: Asignación de etiquetas y distribución de probabilidad.

Estructura del circuito cuántico generativo

El propósito del circuito cuántico generativo es convertir un estado inicial en un estado final cuya medición sea una distribución de probabilidades similar a la distribución objetivo. La estructura del circuito consta de diversas capas de rotaciones intercaladas y de operaciones de entrelazamiento:

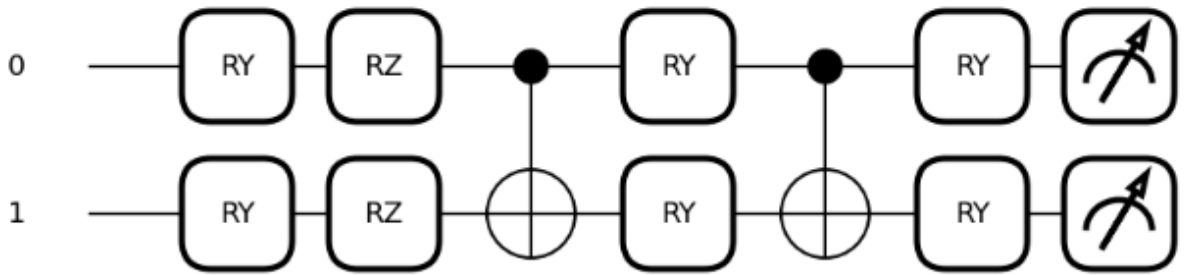


Figura 18: Circuito cuántico para generar caracteres con 2 qubits.

El circuito cuenta con una primera capa de rotaciones en la que se aplican operaciones de rotación en los ejes Y y Z en cada qubit. Estas rotaciones constan de cuatro parámetros (dos por qubit), lo que permite ajustar tanto la amplitud como la fase de cada qubit del estado cuántico.

Tras la primera capa de rotaciones, se aplica una capa de entrelazamiento. Para ello, se utiliza una puerta de control CNOT, donde el primer qubit es el control y el segundo es el objetivo. Así, se consigue establecer una correlación entre los dos qubits. Después de esta capa de entrelazamiento, se implementa otra capa de rotaciones intermedias, en esta ocasión únicamente en el eje Y de cada qubit. Esto posibilita perfeccionar el estado producido después de la primera interacción y optimizar la distribución de las probabilidades objetivo.

Por último, se introduce otra puerta CNOT seguida de una última capa de rotaciones para mejorar la capacidad de aprendizaje del circuito, mejorando así la preparación del estado cuántico antes de la medición.

Función de coste y entrenamiento

Para evaluar el rendimiento del circuito, se mide el estado final y se compara la distribución de probabilidades obtenida con la distribución objetivo. Para comparar ambas distribuciones, se utiliza una función de coste basada en el error cuadrático medio:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (\text{dist}_i - \text{target_probs}_i)^2$$

Donde N es el total de salidas posibles del circuito, en este caso 4, dist_i es la probabilidad generada por el circuito para el estado i , y target_probs_i es la probabilidad objetivo para ese estado. Por lo tanto, si el resultado es bajo, la salida del circuito se acerca a la distribución objetivo [32].

Para el proceso de optimización se utiliza el algoritmo Adam, un método estocástico basado en el descenso del gradiente [33]. El entrenamiento comienza con una elección aleatoria de los 8 parámetros que controlan las puertas de rotación RY y RZ. Estas rotaciones determinan la transformación del estado inicial y por tanto, la distribución de probabilidad resultante tras la medición.

Se realizan varias iteraciones, donde para cada época se calcula la función de coste MSE y se actualizan los parámetros minimizando el coste calculado. Para evitar cambios excesivos en las actualizaciones de los parámetros, se establece la tasa de aprendizaje en 0.005.

Resultados y evaluación

Una vez terminado el entrenamiento, se evalúa el circuito midiendo la distribución de probabilidad en los 4 estados posibles y se compara la probabilidad generada con la probabilidad objetivo. Esto permite comprobar visualmente la calidad del ajuste conseguido por el entrenamiento y ver hasta que punto el circuito ha aprendido a generar la distribución deseada.

En la siguiente tabla se muestran los resultados de los entrenamientos para diferentes números de épocas:

Épocas	Estado “00” (A)	Estado “01” (B)	Estado “10” (C)	Estado “11” (D)
80	0.404	0.301	0.198	0.097
100	0.401	0.300	0.199	0.099
120	0.400	0.300	0.199	0.100
130	0.400	0.300	0.200	0.100

Tabla 2: Distribución generada para diferentes valores de épocas.

En la tabla 2 se observa que a medida que crece el número de épocas, el circuito consigue replicar más fielmente la distribución objetivo. A partir de 100 épocas, el circuito es tan preciso que las diferencias entre la distribución generada y la objetivo son prácticamente nulas, y con 130 épocas, el modelo consigue generar la distribución deseada exactamente. El tiempo de ejecución del experimento completo es de 3 segundos.

5.2.2. Generar secuencias de 2 caracteres con 4 qubits

Extendiendo el experimento anterior, en este se generarán secuencias de dos caracteres en lugar de solamente uno. Para este experimento, se mantienen los 4 símbolos utilizados en el experimento anterior, y la codificación en bitstrings de la secuencia será el resultado de juntar la codificación de ambos símbolos en la tabla 1. Nuevamente, se asigna una distribución de probabilidades para cada estado posible:

Bitstring	Secuencia	Probabilidad objetivo
0000	AA	0.10
0001	AB	0.20
0010	AC	0.05
0011	A_	0.05
0100	BA	0.10
0101	BB	0.05
0110	BC	0.05
0111	B_	0.05
1000	CA	0.05
1001	CB	0.05
1010	CC	0.05
1011	C_	0.05
1100	_A	0.05
1101	_B	0.05
1110	_C	0.05
1111	__	0.05

Tabla 3: Codificación de secuencias de 2 caracteres a partir de bitstrings de 4 bits.

Estructura del circuito cuántico generativo

La estructura del circuito en este experimento es muy similar a la del circuito anterior. Para este experimento es necesario utilizar 4 qubits en lugar de 2, ya que esta vez contamos con 16 posibles estados. Al igual que en el experimento anterior, el circuito comienza con una capa de rotaciones en los ejes Y y Z de cada qubit para ajustar tanto la amplitud como la fase.

Seguido de esta capa de rotaciones, nuevamente vuelve a haber una capa de entrelazamiento entre todos los qubits del circuito, para lo que es necesario usar 3 puertas CNOT creando una cadena.

Por último, antes de la medición, solo fue necesaria una capa de rotaciones en el eje Y para afinar la transformación del estado cuántico y conseguir una distribución de probabilidades muy cercana a la objetivo.

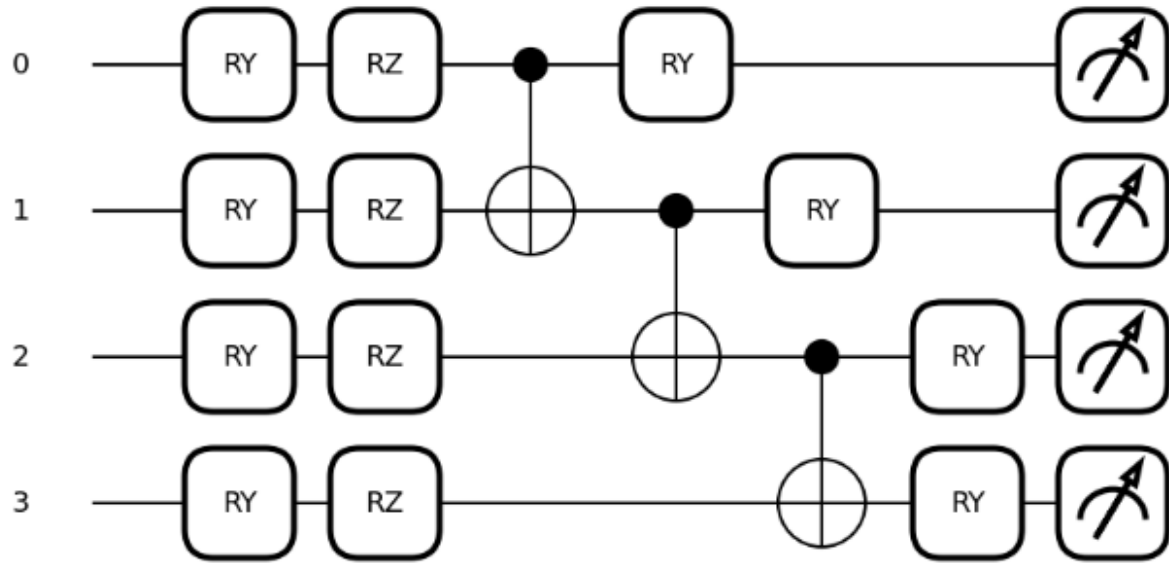


Figura 19: Circuito cuántico para generar cadenas de 2 caracteres.

Función de coste y entrenamiento

Al igual que en el experimento anterior, se utiliza MSE para evaluar el rendimiento del circuito, comparando la distribución de probabilidades generada con la distribución objetivo para las 16 posibles secuencias. Nuevamente durante el entrenamiento se utiliza el optimizador Adam.

Resultados y evaluación

Nuevamente, la métrica de evaluación empleada es comparar la distribución objetivo con la distribución generada.

La siguiente tabla muestra los resultados obtenidos tras un entrenamiento de 1000 épocas, mostrando para cada secuencia el bitstring, la probabilidad generada y la probabilidad objetivo:

Secuencia	Probabilidad Generada	Probabilidad Objetivo
AA	0.094	0.100
CA	0.054	0.050
BA	0.089	0.100
_A	0.059	0.050
AC	0.048	0.050
CC	0.042	0.050
BC	0.044	0.050
_C	0.052	0.050
AB	0.196	0.200
CB	0.048	0.050
BB	0.051	0.050
_B	0.047	0.050
A_	0.044	0.050
C_	0.034	0.050
B_	0.058	0.050
--	0.043	0.050

Tabla 4: Comparación entre la distribución generada y la distribución objetivo para secuencias de 2 caracteres.

Como se observa en la tabla 4, con 1000 épocas de entrenamiento el circuito cuántico es capaz de reproducir la distribución objetivo de manera muy precisa. La ejecución completa del experimento tardó 22 segundos.

5.2.3. Generar secuencias de 3 palabras con 9 qubits

Extendiendo los experimentos anteriores, en este tercero se realizarán secuencias de 3 palabras. Para el experimento se seleccionaron 8 palabras, las cuales se codificaron en bitstring de longitud 3.

Bitstring	Palabra
000	hola
001	mundo
010	amigo
011	casa
100	perro
101	gato
110	sol
111	luna

Tabla 5: Codificación de palabras con bitstrings de 3 bits.

Con 9 qubits, es posible representar $2^9 = 512$ estados diferentes, sin embargo, para este experimento solamente se le asignarán probabilidades a 4 estados:

Secuencia de palabras	Probabilidad Objetivo
hola mundo amigo	0.30
amigo casa perro	0.25
perro gato sol	0.20
sol luna hola	0.25

Tabla 6: Distribución objetivo de probabilidad para secuencias de 3 palabras.

A las otras 508 posibles combinaciones de palabras se les asignó una probabilidad de cero.

Estructura del circuito cuántico generativo

Para este tercer experimento, debido al aumento en el número de estados, la estructura que mejor resultado obtuvo se basó en el template “StronglyEntanglingLayers”, que genera un ansatz variacional mediante 3 capas parametrizadas. Al igual que en los experimentos anteriores, se manipula el estado de los 9 qubits aplicando capas de rotaciones y entrelazamiento buscando transformar el estado inicial en un estado final que replique la probabilidad objetivo para los 512 estados posibles.



Figura 20: Circuito cuántico para generar secuencias de 3 palabras.

Función de coste y entrenamiento

Nuevamente, el proceso de entrenamiento se lleva a cabo utilizando el algoritmo Adam para optimizar los parámetros del circuito y la función de coste se basa en el MSE.

Además, esta vez se utilizaron 1500 épocas y se realizaron reinicios con 5 semillas para seleccionar la que minimiza el coste final.

Resultados y evaluación

Finalmente, después de entrenar el modelo estas son las probabilidades generadas para las secuencias relevantes:

Bitstring	Secuencia	Prob. Generada	Prob. Objetivo
110111000	sol luna hola	0.274	0.250
010011100	amigo casa perro	0.224	0.250
000001010	hola mundo amigo	0.276	0.300
100101110	perro gato sol	0.225	0.200

Tabla 7: Comparación entre la distribución generada y la distribución objetivo para las secuencias relevantes de 3 palabras.

Aunque existen discrepancias, la aproximación es bastante razonable. El tiempo de ejecución del experimento completo fue de aproximadamente 15 minutos, unos 3 minutos por semilla.

5.2.4. Conclusiones y perspectivas del enfoque

Como se ha podido comprobar en estos tres experimentos, las QCBM son capaces de aprender y replicar distribuciones de probabilidad discretas, incluso cuando el número de qubits hace que el espacio de estados crezca exponencialmente. En la rama generativa del NLP, modelos clásicos como GPT hacen justamente esto, predicen las siguientes palabras en base a un contexto según una distribución de probabilidades generada sobre un vocabulario a partir de una red neuronal.

Para obtener una comparativa, se replicó el problema del tercer experimento con una red neuronal clásica que, al igual que la QBCM, tenía el objetivo de aprender y replicar la distribución de probabilidades para los 512 posibles estados. La red se diseñó con una entrada ficticia, una capa oculta de 128 neuronas y una capa de salida con 512 neuronas, seguida de una función Softmax para que la distribución de probabilidad generada sea válida.

Al igual que en los experimentos cuánticos, para la red neuronal clásica, se utilizó MSE como función de coste, el optimizador Adam, una tasa de aprendizaje de 0.005 y un entrenamiento de 1500 épocas. Los resultados y la comparativa se muestran en la siguiente tabla:

Secuencia	Prob. Cuántica	Prob. Clásica	Objetivo
hola mundo amigo	0.276	0.300	0.300
amigo casa perro	0.224	0.249	0.250
perro gato sol	0.225	0.200	0.200
sol luna hola	0.274	0.250	0.250

Tabla 8: Comparación entre la distribución generada por el modelo cuántico, el modelo clásico y la distribución objetivo.

Como se puede observar en la tabla 8, los resultados de la red neuronal clásica son prácticamente perfectos,

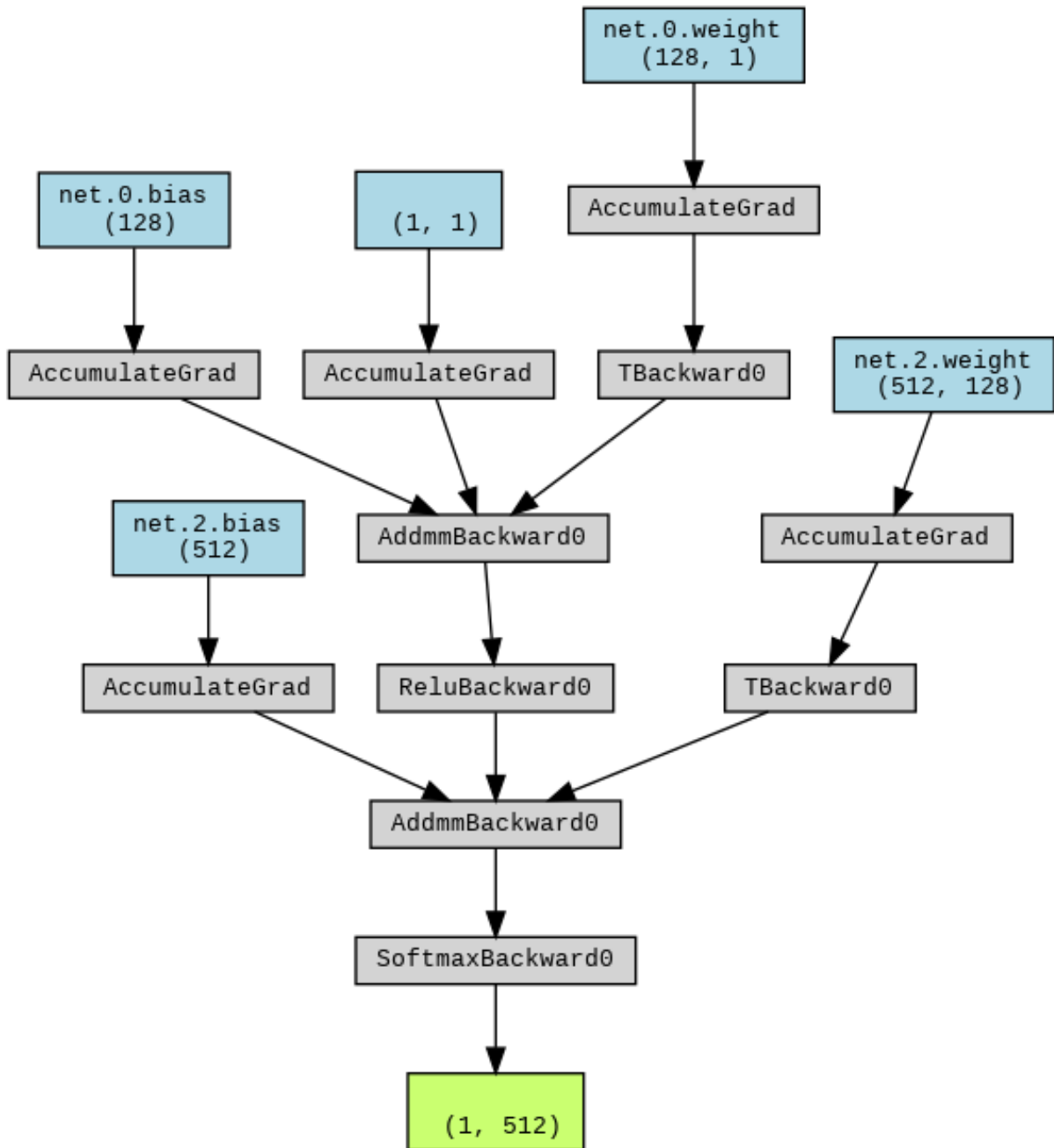


Figura 21: Grafo de cómputo generado con torchviz para la red neuronal clásica.

mientras que el modelo cuántico, aunque es funcional y logra una aproximación considerable, tiene ciertas desviaciones en las probabilidades esperadas, entre 0.024 y 0.026. Además, en términos de velocidad la red neuronal clásica también fue superior, tardando apenas 2 segundos en ejecutarse.

Aunque en este experimento simple la red neuronal clásica supera claramente al modelo cuántico en eficiencia y precisión, el enfoque cuántico sirve como prueba de concepto.

Actualmente, este enfoque enfrenta varias limitaciones. Algunas de las más notables son:

- **Escalabilidad:** El número de qubits crece exponencialmente con el vocabulario del problema, y hoy en día el número de qubits reales disponibles en los ordenadores cuánticos es limitado. Además, en este enfoque en particular el número de qubits también crece con la longitud de la oración.
- **Ruido:** El ruido es otro de los problemas actuales del hardware cuántico. Se llama ruido a las imperfecciones físicas que alteran el comportamiento ideal de un circuito cuántico. Los qubits son extremadamente sensibles al entorno, y cualquier interacción no deseada puede cambiar su estado de forma errónea [34]. Aunque actualmente los ordenadores cuánticos cuentan con algo más de 100 qubits, cuando se usan muchos qubits es posible que el resultado sea afectado por el ruido.
- **Entrenamiento:** Otra de las limitaciones es que por el momento los métodos optimización cuánticos no están tan maduros como los de deep learning.

Sin embargo, esto no quiere decir que las qbcm no tengan ninguna ventaja sobre los métodos clásicos.

5.3. Generación de secuencias mediante Simulated Annealing y SWAP Test

En esta sección exploraremos otro método para generar lenguaje con circuitos cuánticos. En este enfoque, se emplea un algoritmo de búsqueda clásico simulated annealing que recorre el espacio de oraciones candidatas.

5.3.1. Codificación con bitstrings

Para esta primera versión, cada oración se representa como un bitstring de 9 bits, usando las mismas palabras y codificación que en la Tabla 5.

En este enfoque, se utiliza un circuito clasificador basado en SWAP test. El SWAP test es un procedimiento que se utiliza en computación cuántica para comprobar la diferencia entre los dos estados cuánticos [35].

El circuito cuántico utilizado en este experimento cuenta con 19 qubits, de los cuales los últimos 9 sirven para cargar la referencia, los 9 anteriores representan la oración candidata, y el primero de los qubits es un qubit ancilla. Este qubit sirve como controlador en la comparación entre la referencia y el candidato, y no almacena información de la oración directamente, sino que actúa como intermediario durante la ejecución del SWAP test.

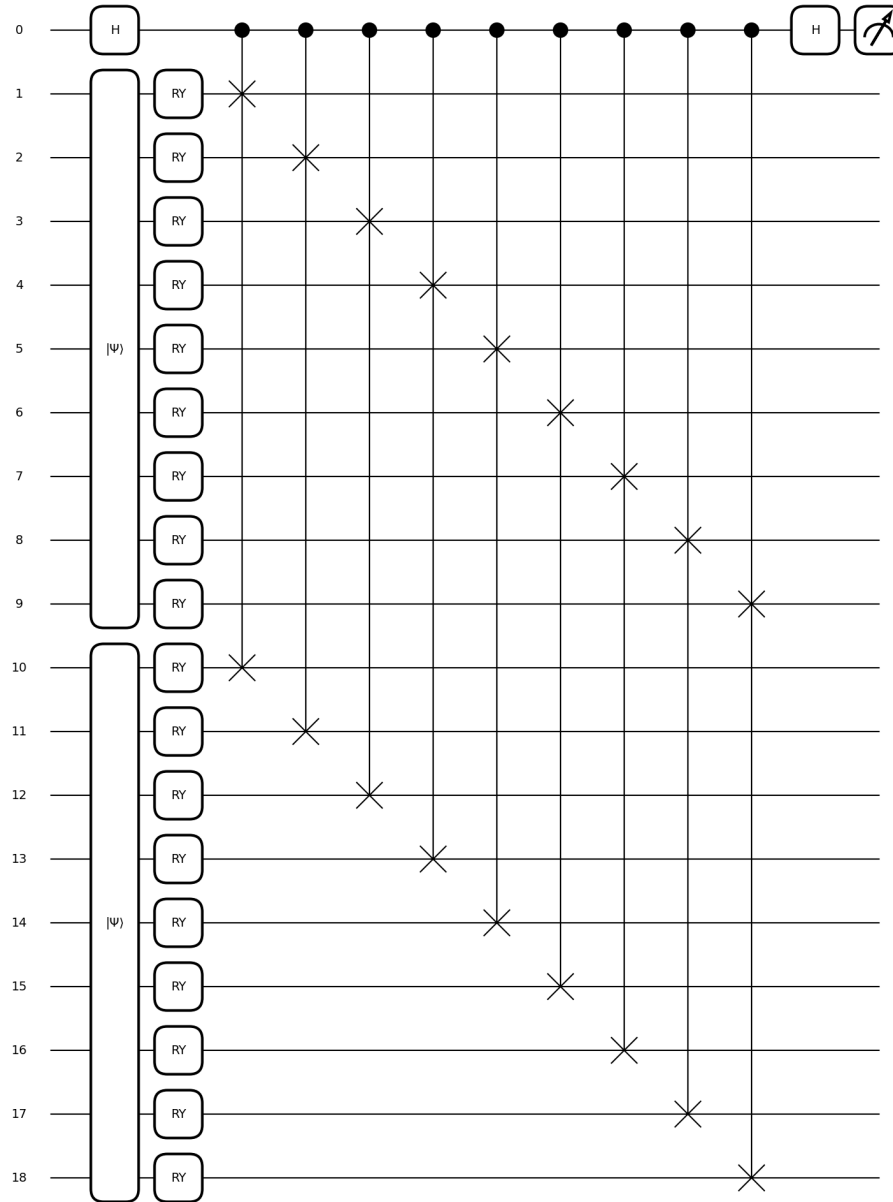


Figura 22: Diagrama del circuito que implementa el SWAP test con los registros candidato y referencia. Codificación por bitstrings.

El funcionamiento del circuito es sencillo, el primer paso es inicializar el circuito. Para ello, se carga una oración de referencia, en este caso “hola perro amigo” que se codifica como 000100010, y otra oración candidata de tres palabras totalmente aleatorias. Por último, el qubit ancilla se inicializa a 0.

Después, se le aplica una puerta Hadamard al qubit ancilla para ponerlo en superposición. Una vez el qubit está en superposición, se aplican las puertas CSWAP conectando los qubits del candidato con los de la referencia. Estas puertas, si el qubit ancilla está en estado $|0\rangle$ no hacen nada, pero si está en estado $|1\rangle$, se intercambian los qubits del candidato y la referencia. No obstante, como el qubit se encuentra en superposición debido a la puerta Hadamard, el circuito hace las dos cosas al mismo tiempo, intercambia todos los registros y a la vez no intercambia ninguno, es decir, se ha generado un estado entrelazado entre el qubit ancilla y los dos registros. Algebraicamente, el sistema en este punto está en el siguiente estado [36]:

$$\frac{1}{\sqrt{2}} (|0\rangle \otimes |\psi\rangle \otimes |\phi\rangle + |1\rangle \otimes |\phi\rangle \otimes |\psi\rangle)$$

Donde:

- $|\psi\rangle$ es el estado cuántico del candidato
- $|\phi\rangle$ es el estado cuántico de la referencia

Al poner la puerta Hadamard del final, el estado anterior pasa a ser una superposición con interferencia que depende de cuánto de parecidos son $|\psi\rangle$ y $|\phi\rangle$. Los pasos algebraicos completos pueden consultarse en [36], pero lo importante es que al final:

$$P(\text{medir } 0) = \frac{1}{2} + \frac{1}{2} |\langle\psi|\phi\rangle|^2$$

Por lo tanto, si $|\psi\rangle = |\phi\rangle$, entonces:

- $|\langle\psi|\phi\rangle|^2 = 1$
- $\Rightarrow P(0) = 1$

Es decir, la probabilidad de medir 0 en el qubit ancilla está directamente relacionada con la similitud entre el candidato y la referencia, cuanto más similares sean más probable será medir 0, mientras que si son muy distintos la probabilidad de medir 1 es mayor.

En caso de que la probabilidad de medir 0 no sea de al menos 0.95, se modifica el candidato mediante Simulated Annealing [37]. SA es el algoritmo de optimización encargado de inicializar la oración candidata aleatoriamente y de modificarla mediante la alteración de una palabra en cada iteración hasta conseguir que la probabilidad de medir 0 sea de al menos 0.95.

Es importante aclarar que en un ordenador cuántico real, al medir como en el circuito de este experimento se obtendría únicamente un $|0\rangle$ o un $|1\rangle$, por lo tanto, habría que realizar varias mediciones para estimar la probabilidad. Sin embargo, como el experimento se ha ejecutado en un simulador, es posible acceder directamente a la probabilidad.

La ejecución del experimento duró 24 segundos, y fue capaz de generar la frase deseada “hola perro amigo” con una probabilidad de 0.9975 en 54 iteraciones.

Aunque el método es efectivo para representar palabras, nuevamente tiene un problema de escalabilidad relacionado con el tamaño del vocabulario y la longitud de la frase.

A modo de ejemplo, podríamos representar todas las palabras del diccionario español con 17 qubits. Para un circuito como el de nuestro experimento, harían falta 17 qubits para preparar el estado del candidato, otros 17 qubits para preparar el estado de referencia y 1 último qubit ancilla para controlar las compuertas CSWAP durante el SWAP test. En total se requerirían 35 qubits. Actualmente 35 qubits es un número bastante factible para la tecnología de la que disponemos, sin embargo, de esta forma solo podríamos trabajar con una única palabra. Para una oración de 3 palabras como la del ejemplo se requerirían $3 \times 17 = 51$ qubits para el candidato, lo mismo para la referencia y un qubit ancilla, en total 103 qubits, lo que, aunque posible a día de hoy, ya que los ordenadores cuánticos de IBM cuentan con alrededor de 130 qubits, deja de ser tan factible teniendo en cuenta el ruido generado por utilizar tantos qubits.

5.3.2. Codificación por fase

Con el objetivo de mejorar el problema de escalabilidad del experimento anterior, en este segundo se probó a codificar las palabras como ángulos en radianes. Cada palabra se codifica aplicando primero una puerta Hadamard para poner el qubit en superposición y luego una puerta R_Z que aplica el ángulo correspondiente. Los ángulos con los que se codifica cada palabra en este experimento son los siguientes:

Palabra	Ángulo
hola	0
mundo	$\frac{\pi}{4}$
amigo	$\frac{\pi}{2}$
casa	$\frac{3\pi}{4}$
perro	π
gato	$\frac{5\pi}{4}$
sol	$\frac{3\pi}{2}$
luna	$\frac{7\pi}{4}$

Tabla 9: Codificación del vocabulario mediante ángulos en radianes.

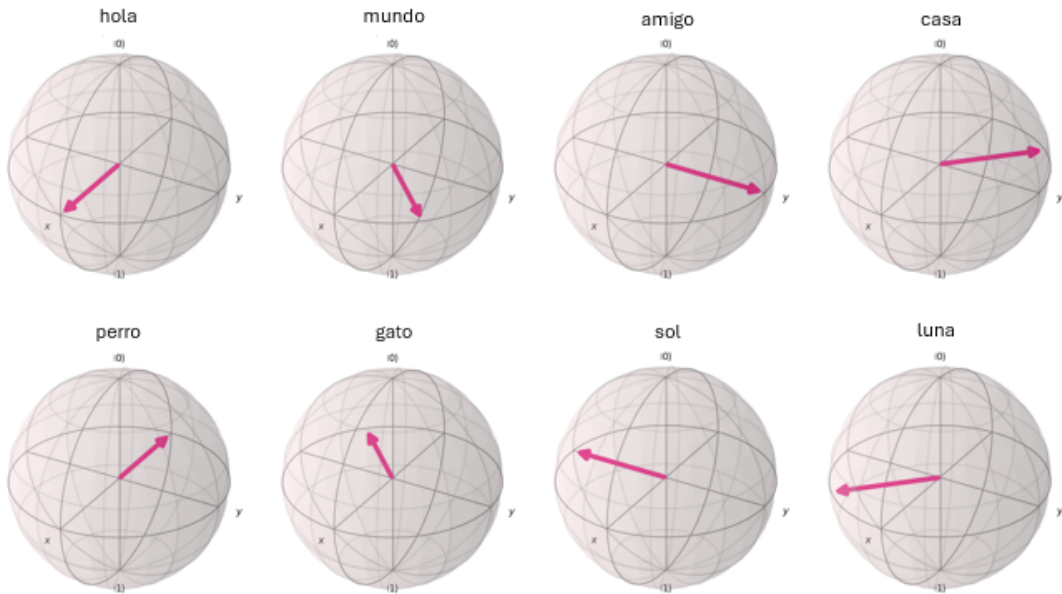


Figura 23: Visualización del vocabulario codificado por fase en la esfera de bloch.

El teorema de Holevo dice que aunque un estado cuántico puede codificar más información en superposición, al medirlo no se pueden obtener más de n bits clásicos a partir de n qubits [38]. Por lo tanto, en este experimento, en lugar de codificar mediante bits, se aprovecha la capacidad que tienen los qubits de poder almacenar información en su fase por naturaleza, lo que reduce considerablemente el número de qubits requeridos en el circuito.

El funcionamiento del circuito es igual que en el experimento anterior, ya que solo ha cambiado la forma de codificar las palabras. Sin embargo, este cambio provoca que solo sea necesario utilizar 1 qubit para cada palabra de la oración, por lo que ahora son necesarios 7 qubits para trabajar con la oración “hola perro amigo”, 3 para el candidato, 3 para la referencia y el ancilla.

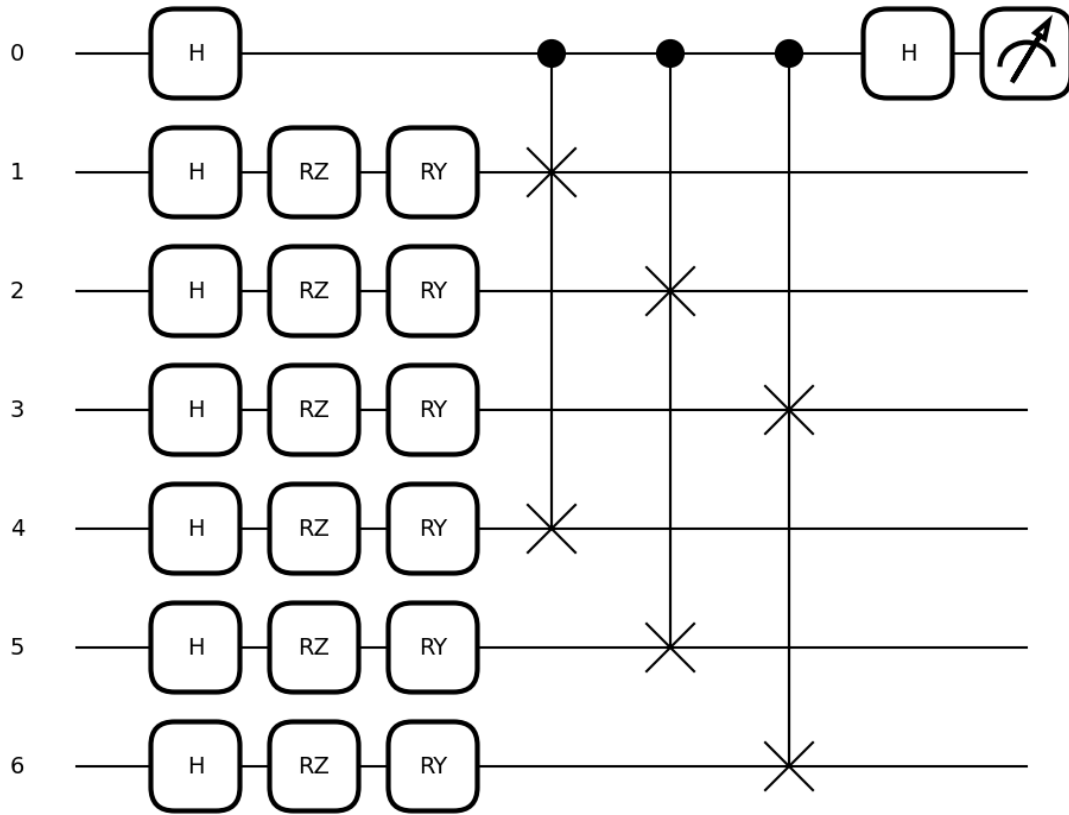


Figura 24: Diagrama del circuito que implementa el SWAP test con los registros candidato y referencia. Codificación por fase.

El experimento tardó menos de un segundo en ejecutarse, y consiguió generar la oración “hola perro amigo” en 63 iteraciones con 0.9983 de probabilidad.

Teóricamente, un qubit puede representar infinitos valores, ya que su fase es continua. Esto significa que, teóricamente, es posible codificar un vocabulario infinito asignando a cada palabra un ángulo distinto en el espacio de Hilbert. De esta forma, el número de qubits del circuito aumentaría en función de la longitud de la secuencia de palabras, pero no dependería del tamaño del vocabulario.

Para comprobar esta teoría y ver sus limitaciones, se realizó un experimento como el anterior pero con un vocabulario de 15 palabras, y con el objetivo de generar secuencias de longitud 5. Al igual que en el experimento anterior, cada palabra se codificó con un ángulo equidistante entre 0 y 2π .

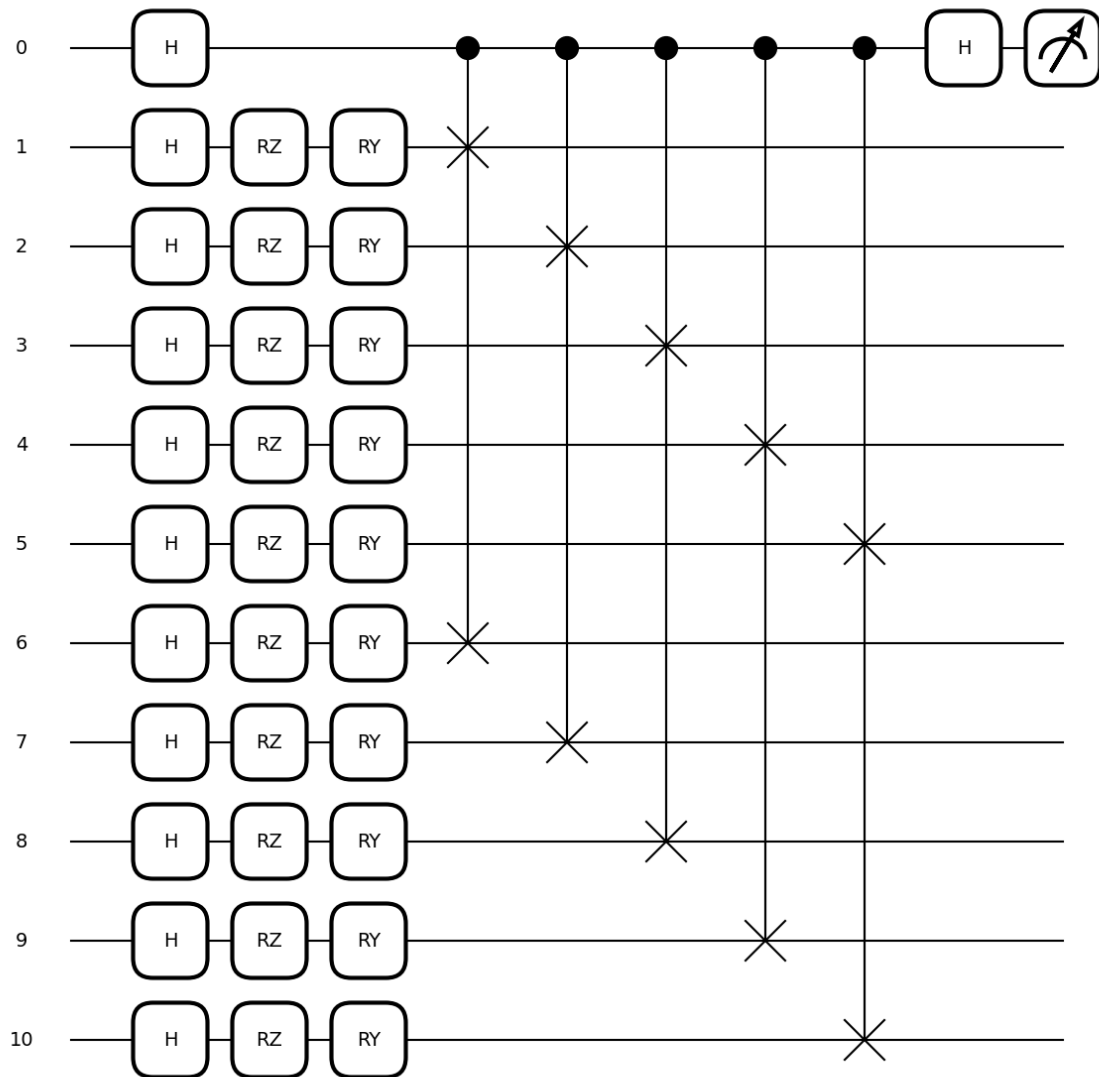


Figura 25: Diagrama basado en SWAP test para secuencias de 5 palabras

Como se puede observar en la Figura 25, a pesar de que el problema creció tanto en vocabulario como en longitud, el circuito solo requirió 4 qubits extra, dos para la referencia y dos para el candidato, debido a que la longitud de la secuencia aumentó 2 palabras.

La secuencia elegida para probar el experimento fue “hola perro amigo casa sol”, sin embargo, la secuencia candidata aceptada por el circuito después de 77 iteraciones fue “estrella casa mundo amigo sol”, con una probabilidad de medir 0 de 0.9641. A pesar de que son oraciones muy diferentes.

Probablemente, el motivo por el cual el algoritmo acepta la secuencia “estrella casa mundo amigo sol” se deba a que los ángulos asignados a ciertas palabras están relativamente cerca en el espacio de Hilbert. Recordemos que el SWAP test no busca que las secuencias sean exactamente iguales, sino que acepta cualquier secuencia cuya representación cuántica obtenida mediante la codificación por fase tenga un alto solapamiento con la representación del estado de referencia.

Este experimento demuestra que con una codificación por fase, aunque el tamaño del vocabulario no afecta directamente al número de qubits ni a la complejidad estructural del circuito, sí influye en el rendimiento global. Esto se debe a que, a medida que se incrementa el tamaño del vocabulario, se deben asignar ángulos a un mayor número de palabras dentro de un mismo rango y como consecuencia, los estados cuánticos correspondientes pueden volverse muy cercanos entre sí, lo que dificulta su discriminación a través de mediciones estándar.

5.3.3. Codificación por amplitud y fase con FastText

Este experimento se propuso para mejorar el anterior. Los dos experimentos tienen la misma base, pero en este segundo, además de codificar utilizando la fase, también se utilizó la amplitud. De esta forma, se aprovecha toda la esfera de Bloch, permitiendo codificar en 2 dimensiones y aumentando así la distancia entre los ángulos de las diferentes palabras.

Además, en lugar de utilizar ángulos equidistantes en la codificación, se utiliza FastText para asignar los ángulos en función de la similitud entre los significados de las palabras. FastText es un modelo de embeddings de palabras preentrenado desarrollado por Facebook AI [39]. Por poner un ejemplo, según FastText, la similitud entre “dog” y “cat” es de un 0.71, mientras que la similitud entre “dog” y “moon” es de 0.18. Se espera que gracias a ese método de codificación con ayuda de FastText, las palabras codificadas que sean similares se encuentren más cerca que las que tengan significados distintos en el espacio de Hilbert, lo que podría mejorar el rendimiento.

Para codificar el vocabulario, primero se calcularon las similitudes entre todas las parejas de palabras, y se creó una matriz de similitud en función de los resultados. Para calcular la distancia en función de la similitud se aplicó la fórmula: $Distancia = 1 - similitud$. Una vez calculadas las distancias, se aplicó Multidimensional Scaling para reducir la dimensionalidad a 2D, y por último, se mapearon las coordenadas obtenidas en la esfera de Bloch.

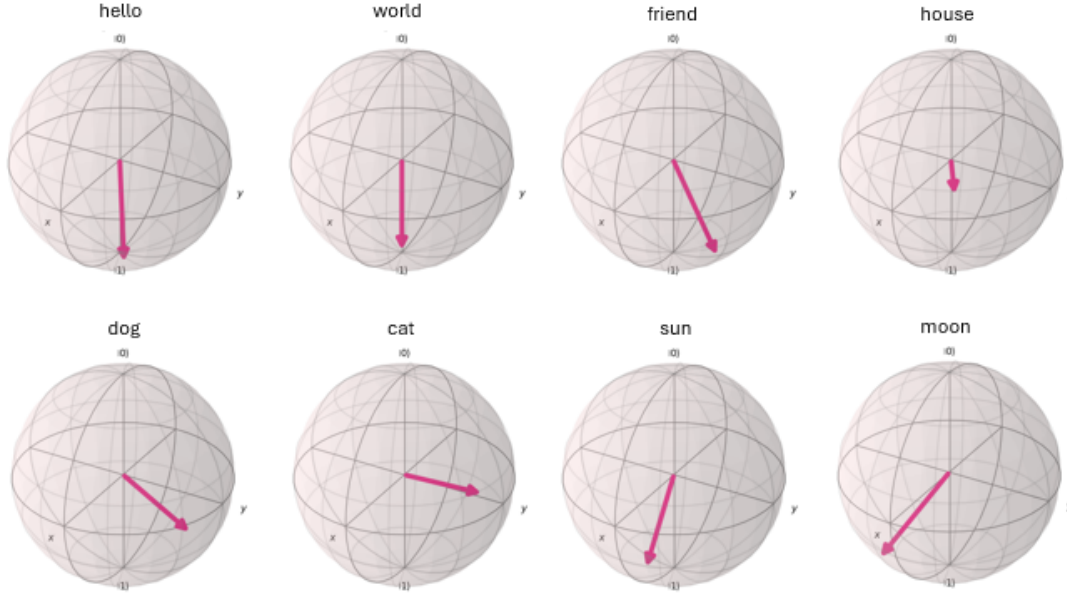


Figura 26: Visualización de 8 palabras codificadas por amplitud y fase según la similitud en FastText.

Se enfrentó este nuevo experimento al reto anterior de codificar 15 palabras y representar oraciones de longitud 5 utilizando 11 qubits, sin embargo, el resultado no fue el esperado. Para la oración de referencia “hello dog friend house sun”, la oración candidata aceptada por el circuito después de 45 iteraciones fue “star fire friend water sun”, con un 0.9659 de probabilidad. Esto ocurre porque a pesar de que las palabras similares están más separadas en el espacio de Hilbert, la realidad es que todas están relativamente cerca, por lo que el SwapTest las reconoce como palabras similares.

El modelo de FastText usado en los experimentos utiliza 300 dimensiones para representar cada palabra en su espacio vectorial. En este experimento en cambio, solo se han utilizado 2 dimensiones, lo que es una reducción demasiado grande que no permite representar fielmente la distancia entre palabras.

En teoría, es posible añadir más qubits para aumentar la dimensionalidad con la que se representan las palabras. Se realizaron experimentos con el dataset bbc-news donde cada palabra estaba representada por 2 qubits en lugar de 1, utilizando así 4 dimensiones. Aún así, la dimensionalidad sigue sin ser suficiente, es necesario utilizar más qubits y el objetivo de esta codificación era reducir el número de qubits necesarios en la codificación por bitstring sin afectar drásticamente al resultado de las oraciones.

5.4. Transformer con auto-atención cuántica

La auto-atención, es un componente esencial de los transformers clásicos, ya que asigna pesos a los distintos tokens de una secuencia para resaltar cuáles son más relevantes para predecir la siguiente palabra. En este apartado, se describirá y se probará un mecanismo de auto-atención empleando circuitos cuánticos. Este bloque de auto-atención cuántica, se inspira en el artículo de Dynex “How to Implement a Quantum Self-Attention Transformer on Dynex” [40].

La Transformada Cuántica de Fourier es la versión cuántica de la transformada discreta de Fourier tradicional. En computación cuántica, la QFT actúa sobre un estado de qubits y lo transforma a una superposición de componentes de frecuencia [41].

La QFT inversa es la operación inversa de la transformada cuántica de Fourier, convierte una señal de nuevo al dominio del tiempo después de haberla llevado al dominio de la frecuencia [41].

El difusor de Grover, es un circuito cuántico basado en el algoritmo de Grover. En Grover, primero se “marca” el estado deseado mediante un oráculo, cambiándolo de fase, luego el difusor incrementa la amplitud de ese estado marcado y reduce las de los demás. El objetivo es ir incrementando la amplitud del estado deseado y reduciendo la de los demás estados, de forma que tras repetir el proceso varias veces, el estado deseado sea el que más probabilidades tiene de ser medido [42].

5.4.1. Primera versión

A continuación se presenta la primera versión ejecutada del experimento, que se realizó siguiendo los pasos indicados en [40].

El funcionamiento del circuito cuántico es el siguiente:

- En primer lugar, antes de comenzar con el circuito, se utiliza Word2Vec [43] para convertir las palabras en vectores. Word2Vec entrena una pequeña red neuronal a partir de un corpus de texto para aprender asociaciones entre las palabras. Similar a como se ha hecho en experimentos anteriores, el objetivo es traducir el lenguaje a una forma numérica que preserve relaciones semánticas.
- Las puertas iniciales del circuito son la preparación y codificación inicial antes de realizar la atención cuántica. Estas puertas crean un estado cuántico complejo y entrelazado que refleja las relaciones semánticas de las palabras en función de los vectores resultantes del Word2Vec.
Las puertas RX, RY y RZ rotan el estado del qubit alrededor de los ejes X, Y y Z respectivamente. Cada ángulo de rotación se basa directamente en los valores numéricos de los embeddings, de manera que cada qubit representa una dimensión específica del vector que representa la palabra.
Las puertas CRZ realizan rotaciones en el eje Z de un qubit controlado por otro qubit. Esto permite que cada dimensión del vector interactúe y afecte a otras, generando una representación más rica y entrelazada de la información semántica.
Las puertas CNOT introducen entrelazamiento cuántico entre los qubits, algo fundamental para el mecanismo de auto-atención cuántica ya que permite hacer que diferentes dimensiones o palabras tengan dependencia entre sí.
- La siguiente parte del circuito aplica la QFT seguida de su inversa. Aunque en un principio esta parte del circuito pueda parecer redundante, en computación cuántica, esta secuencia en combinación con las puertas previas, genera interferencias específicas que reorganizan el estado cuántico y permite preparar la superposición para el siguiente paso.
- Una vez han surgido estas interferencias, el difusor de Grover realiza una inversión sobre la media de las amplitudes. El objetivo es resaltar aquellas configuraciones marcadas implícitamente por el estado cuántico previo, simulando así el proceso de “atención”. La consecuencia es que las palabras más relevantes ven incrementada su amplitud, y por tanto, su probabilidad de ser medidas al final del circuito.
- Por último, las puertas finales tienen el objetivo de preparar el estado final para la medición. Tras estas puertas, el estado resultante es medido. Al medir, se obtienen valores que se procesan posteriormente con una función Softmax, resultando en una distribución de probabilidades sobre las posibles palabras, de la cual escogemos la más probable como siguiente palabra.
- El circuito incluye un paso llamado BasisEmbedding tanto al principio como al final, que convierte el vector numérico en estados cuánticos. El primer $|\psi\rangle$ corresponde al embedding original que se introduce al circuito y el último $|\psi\rangle$ es una re-codificación que ayuda a que el estado final esté claramente relacionado con el embedding inicial, facilitando la interpretación directa del resultado final.

Para ejecutar el circuito, se creó un dataset de 20 frases cortas en castellano relacionadas con inteligencia artificial y computación cuántica, como por ejemplo “La superposición cuántica permite nuevas posibilidades” o “La visión por computador usa redes neuronales”.

Después de tokenizar el dataset, se entrena el modelo Word2Vec para generar embeddings de 8 dimensiones que representan cada palabra del vocabulario.

Por último, se selecciona una oración de entrada. Los embeddings de palabras de la oración seleccionada son la entrada del circuito cuántico, cuyo funcionamiento se ha explicado anteriormente. Es importante aclarar que en esta primera versión, la entrada del circuito es cada una de las palabras de la oración seleccionada, de forma individual, y el circuito genera un output para cada palabra de entrada. Esto quiere decir que en este primer experimento la atención no es multi-token, ya que cada embedding se procesa por separado, sin comparar entre tokens. La parte de atención se realiza dentro de cada embedding mediante el entrelazamiento entre dimensiones.

Frase de entrada	Texto generado
El quantum computing promete grandes cambios	en es cuántico cuántico de cuántico
La atención es clave en transformadores de lenguaje	cuántico cuántico cuántico es se en la es
El futuro de la computación es paralelo y cuántico	en el la cuántico cuántico cuántico es es es
La inteligencia artificial aprende patrones complejos	cuántico es es cuántico cuántico es
El hardware cuántico evoluciona rápidamente	en el es cuántico cuántico

Tabla 10: Resultados de texto generado a partir de distintas frases de entrada.

Como se observa en la tabla 10, las salidas del circuito no son coherentes gramaticalmente. La palabra “cuántico” es predicha la mayor parte del tiempo, seguido de diferentes stopwords. Esto puede deberse principalmente a que el vocabulario es pequeño y contiene mayoritariamente oraciones relacionadas con la cuántica. Como el argmax elige siempre la probabilidad máxima, podría sesgar las salidas hacia las palabras más frecuentes en el dataset.

El hecho de que el circuito procesa cada token de la secuencia de forma independiente también es una limitación importante, ya que no existe un mecanismo que modele dependencias inter-token.

Otra posible limitación podría ser la dimensión de embedding, con 8 dimensiones es posible que distintas palabras se proyecten en patrones de probabilidad muy similares.

A pesar de las limitaciones, el hecho de que la mayoría de predicciones sean “cuántico”, también podría indicar que el circuito es capaz de captar el tema más importante del dataset sin lograr un refinamiento sintáctico. Sin embargo, esta última conclusión aún no está clara.

5.4.2. Segunda versión

Para intentar mejorar el rendimiento del primer experimento, se intentó aumentar tanto el tamaño del dataset como la dimensionalidad del circuito, sin embargo, al poner más de 12 dimensiones, no era posible lanzar las ejecuciones en el simulador debido a problemas con la capacidad de la RAM, ya que el tamaño del espacio de Hilbert crece exponencialmente con el número de qubits, por lo que se requiere una cantidad de memoria que crece como 2^n complejos para simular n qubits. Por ejemplo, usando 12 qubits, es necesario almacenar $2^{12} = 4096$ amplitudes complejas, si usamos 16, el número asciende a $2^{16} = 65536$, lo que supone un incremento notable en el consumo de memoria. Debido a esto, los simuladores clásicos se vuelven inviables con un número elevado de qubits.

Para poder ejecutar el experimento, se decidió que lo mejor sería lanzarlo en un ordenador cuántico real de IBM.

Al intentar lanzar el código en hardware real mediante la API de IBM Quantum [44], surgían incidencias técnicas de compatibilidad con los backends de IBM y con la transpilación, por lo que se decidió pasar el código de PennyLane a Quiskit [45]. Esta decisión se tomó porque Quiskit está desarrollado directamente por IBM, por lo que la compatibilidad y optimización con sus dispositivos cuánticos están asegurados. Además, Quiskit permite un control directo y manual sobre la transpilación del circuito según las restricciones específicas del hardware real.

El proceso seguido para lanzar el código en un ordenador cuántico real fue el siguiente:

- Primero fue necesario realizar una conexión con IBM mediante el token de autenticación que se proporciona al crear la cuenta.
- Una vez establecida la conexión, es necesario seleccionar el dispositivo cuántico en el cuál lanzar la conexión. En este caso, se programó para seleccionar automáticamente el dispositivo con menos cola de ejecución.
- Antes de ejecutar el circuito, se adapta al hardware real elegido mediante un proceso llamado transpilación. La transpilación es el proceso de transformar el circuito cuántico de alto nivel escrito en Quiskit a una versión equivalente que sea compatible con el ordenador cuántico real seleccionado. Cada ordenador cuántico tiene una arquitectura diferente según su conjunto limitado de puertas, restricciones en la conectividad entre qubits, tiempos de coherencia limitados, y niveles variables de ruido. Durante la transpilación se realizaron distintas operaciones, entre ellas la descomposición de puertas complejas en puertas primitivas compatibles con el ordenador cuántico, inserción de operaciones adicionales como SWAPs para ajustar el circuito a la conectividad disponible entre los qubits físicos, y optimización general del circuito, buscando minimizar errores causados por conexiones limitadas entre los qubits reales y evitar operaciones innecesarias reduciendo el tiempo de ejecución.
- Ya con el circuito adaptado, se definieron los observables para medir los qubits. Se seleccionaron operadores tipo Pauli Z. Estos observables, devuelven un valor entre -1 y +1 para cada qubit del circuito. Este valor obtenido es la probabilidad relativa del qubit de encontrarse en los estados $|0\rangle$ o $|1\rangle$. Un valor cercano a +1 indica mayor probabilidad de medir $|0\rangle$, mientras que un valor cercano a -1 indica mayor probabilidad de medir $|1\rangle$. Los resultados obtenidos de las mediciones, forman un vector de dimensión tan grande como qubits tenga el circuito. Después, al igual que con los simuladores, a este vector se le aplica la función softmax para convertirlo en una distribución de probabilidades, y luego se aplica un argmax sobre esa distribución para elegir la palabra más probable del vocabulario.
- Por último, realizar la ejecución final se utilizó la herramienta proporcionada por IBM Quantum Runtime EstimatorV2. Esta herramienta ejecuta el circuito optimizado en el hardware real con múltiples repeticiones. Además, también aplica técnicas automáticas de mitigación de errores para mejorar la precisión de los resultados.

Primera ejecución

La primera ejecución en hardware real fue con la misma dimensionalidad y dataset que en el simulador, para comprobar si existía algún tipo de diferencia. Este experimento se ejecutó en el ordenador `ibm_sherbrooke`.

ibm_sherbrooke

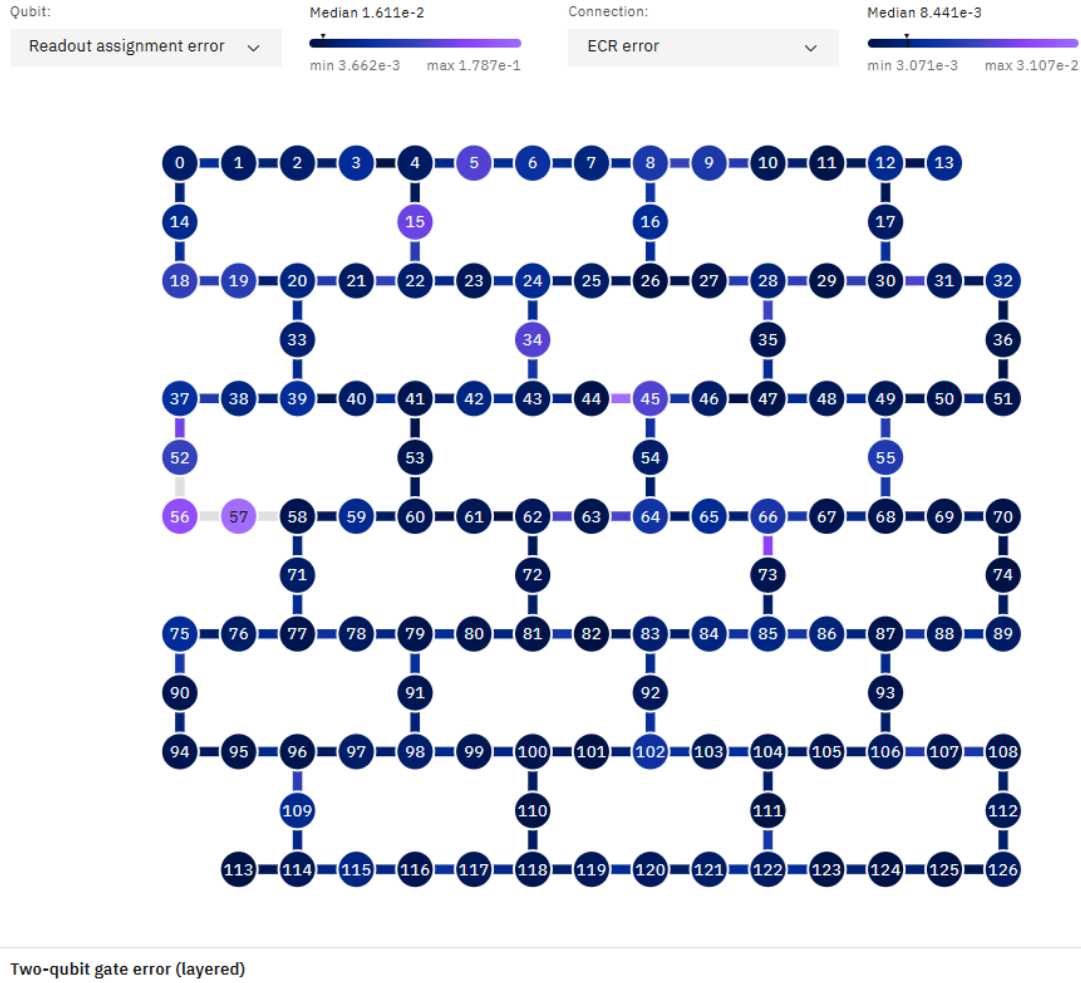


Figura 28: Topología y errores del dispositivo ibm_sherbrooke [5].

En la Figura 28 se muestra la topología del ordenador cuántico real utilizado. Los nodos representan los qubits del ordenador y cada línea es una conexión entre qubits. Si dos qubits no están conectados mediante una línea, para hacer operaciones entre ellos habría que añadir SWAPs, que pueden introducir más errores. El número dentro de los círculos es el índice del qubit, en este caso van de 0 a 126, ya que el ordenador tiene 127 qubits. El color indica el nivel de error, cuanto más oscuro está el qubit indica un menor error, por ejemplo, qubits como el 56 o el 57 tienen errores de lectura bastante más altos que los demás.

El resultado de la primera ejecución no consigue mejorar a las ejecuciones del simulador. Esto era de esperar ya que el simulador estaba configurado para trabajar sin ruido, mientras que en el ordenador cuántico real es inevitable enfrentar algo de ruido. La oración de entrada fue “La inteligencia artificial aprende patrones” y la oración generada a partir de esta entrada fue “es se de se se”.

<input type="checkbox"/>	czdxncntp6g008hznzg	✔ Completed	20 Mar 2025	20 Mar 2025	20s	Job	ibm_sherbrooke
<input type="checkbox"/>	czdxn2mr3jrg008ph2b0	✔ Completed	20 Mar 2025	20 Mar 2025	19s	Job	ibm_sherbrooke
<input type="checkbox"/>	czdxmntb7tt0008gs0kg	✔ Completed	20 Mar 2025	20 Mar 2025	20s	Job	ibm_sherbrooke
<input type="checkbox"/>	czdxmc1r3jrg008ph280	✔ Completed	20 Mar 2025	20 Mar 2025	15s	Job	ibm_sherbrooke
<input type="checkbox"/>	czdxn2ggaq0008cw6bg	✔ Completed	20 Mar 2025	20 Mar 2025	14s	Job	ibm_sherbrooke

Figura 29: Historial de ejecuciones de la primera prueba[5].

Como se observa en la figura 29, cada palabra predicha supone una ejecución del ordenador cuántico, en

este caso, 5 ejecuciones para 5 palabras predichas. Las ejecuciones tardaron entre 14 y 20 segundos.

```

◆ Palabra 1/5: 'la'
  Resultado obtenido: [ 0.01042969  0.00202923  0.00514605  0.01368967 -0.04338172 -0.0215444
  0.04709517  0.04500246]
  Palabra generada: es

◆ Palabra 2/5: 'inteligencia'
  Resultado obtenido: [ 0.01282324  0.01649638  0.01623585  0.05144513 -0.01503309  0.00050901
  0.00424181  0.00719047]
  Palabra generada: se

◆ Palabra 3/5: 'artificial'
  Resultado obtenido: [-0.00864994  0.02605825 -0.00310244  0.01072786  0.02794864  0.02230033
  0.02298684  0.02362879]
  Palabra generada: de

◆ Palabra 4/5: 'aprende'
  Resultado obtenido: [ 0.01025603  0.00775954  0.00963526  0.05861803  0.01758828 -0.00476064
 -0.00359118  0.01325731]
  Palabra generada: se

◆ Palabra 5/5: 'patrones'
  Resultado obtenido: [ 0.00359456  0.00154819  0.0238464  0.05492819  0.03518545 -0.002122
  0.00308105  0.04116838]
  Palabra generada: se

```

Figura 30: Resultados completos de la ejecución. Visualizado en Google Colab.

En la figura 30 se muestran todos los inputs y outputs del proceso. Como se ha explicado anteriormente, cada palabra es un input y una ejecución independiente. Para cada palabra se obtiene un valor entre $[-1, 1]$ por cada qubit del circuito, número que está directamente relacionado con la dimensionalidad del problema, en este caso 8 qubits para 8 dimensiones. Estos 8 valores obtenidos corresponden al vector de valores esperados $\langle Z \rangle$, que, como se ha explicado anteriormente, se normalizan para obtener una distribución de probabilidad mediante la función softmax y se selecciona la palabra con mayor probabilidad mediante la operación argmax.

Segunda ejecución

Una vez comprobado que en igualdad de tamaño la ejecución en hardware real no mejora la del simulador debido al ruido, es momento de hacer una prueba en la que el hardware real supera al simulador. Para esta prueba, simplemente se duplicó el tamaño del dataset y la dimensionalidad. Es decir, en esta ocasión, se volvió a ejecutar el experimento pero con 40 frases cortas y 16 qubits. El resto del código permaneció igual salvo el Word2Vec, que fue necesario reentrenarlo para 16 dimensiones.

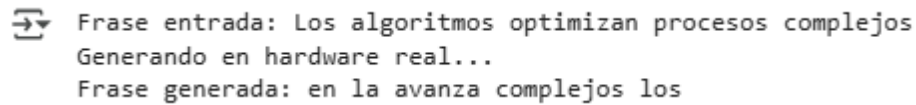
Al duplicar la dimensionalidad el espacio vectorial es más rico, produciendo menos colisiones entre palabras y permitiendo una mayor distinción entre palabras similares. Además, cada qubit adicional duplica la capacidad expresiva del circuito. Con 8 qubits, hay $2^8 = 256$ amplitudes complejas que el circuito puede moldear, en cambio, con 16 qubits, la capacidad crece a $2^{16} = 65536$ amplitudes.

Por otro lado, al duplicar el dataset, además de aumentar el vocabulario de palabras diferentes, Word2Vec recibe el doble de ejemplos por lo que en teoría debería crear vectores más estables y sesgarse menos hacia palabras dominantes.

Al intentar ejecutar un problema de estas características en un simulador sin ruido de PennyLane, utili-

zando Google Colab con configuración de alta capacidad de RAM (51 GB), la ejecución no logró terminar por falta de RAM.

El experimento se ejecutó en el ordenador cuántico `ibm_kyiv`.



↔ Frase entrada: Los algoritmos optimizan procesos complejos
Generando en hardware real...
Frase generada: en la avanza complejos los

Figura 31: Resultado de la segunda ejecución

Como se observa en la Figura 31, la oración generada supera notablemente a la primera prueba. A pesar de que la frase generada sigue sin tener sentido gramatical, hay más variedad de palabras. Además, se observa que en el cuarto token, cuando el input es “procesos” la salida generada es “complejos”, que es la siguiente palabra de la oración de entrada.

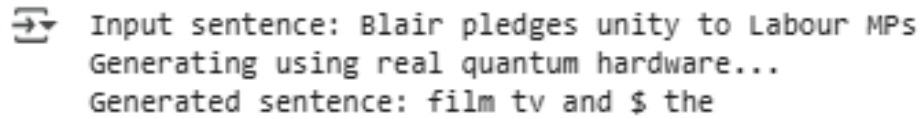
En la Figura 32 se observa que a pesar de aumentar la dimensionalidad y el vocabulario, el tiempo de ejecución no aumenta en exceso, estando entre 19 y 20 segundos por palabra.

<input type="checkbox"/>	cze07bdb7t0008gsb8g	Completed	20 Mar 2025	20 Mar 2025	19s	Job	ibm_kyiv
<input type="checkbox"/>	cze06xb4spc00087g3v0	Completed	20 Mar 2025	20 Mar 2025	20s	Job	ibm_kyiv
<input type="checkbox"/>	cze06f9b7t0008gsb60	Completed	20 Mar 2025	20 Mar 2025	20s	Job	ibm_kyiv
<input type="checkbox"/>	cze06284spc00087g3rg	Completed	20 Mar 2025	20 Mar 2025	20s	Job	ibm_kyiv
<input type="checkbox"/>	cze05jphfwp00088jde0	Completed	20 Mar 2025	20 Mar 2025	20s	Job	ibm_kyiv

Figura 32: Historial de ejecuciones de la segunda prueba[5].

Tercera ejecución

Para la tercera ejecución se utilizó el dataset BBC News Archive [31], que contiene 2225 titulares cortos de noticias en inglés. Además, la dimensionalidad del problema se aumentó a 32 dimensiones.



```
➡ Input sentence: Blair pledges unity to Labour MPs
Generating using real quantum hardware...
Generated sentence: film tv and $ the
```

Figura 33: Resultado de la tercera ejecución

Como se observa en la figura 33, los resultados siguen sin tener coherencia gramatical. Además, en la figura 34, se puede observar que en esta ocasión el aumento en el dataset y la dimensionalidad sí que tuvo un impacto en los tiempos de ejecución, tardando entre 55 y 57 segundos en generar cada palabra.

Este tiempo de ejecución es una limitación ya que durante el desarrollo del proyecto sólo se cuenta con 10 minutos mensuales de ejecución en hardware real.

<input type="checkbox"/>	czema70qadq0008cz1c0	✔ Completed	21 Mar 2025	21 Mar 2025	55s	Job	ibm_sherbrooke
<input type="checkbox"/>	czem9444spc00087jmsg	✔ Completed	21 Mar 2025	21 Mar 2025	55s	Job	ibm_sherbrooke
<input type="checkbox"/>	czem8104spc00087jimg	✔ Completed	21 Mar 2025	21 Mar 2025	56s	Job	ibm_sherbrooke
<input type="checkbox"/>	czem6yktp60g008hrw8g	✔ Completed	21 Mar 2025	21 Mar 2025	55s	Job	ibm_sherbrooke
<input type="checkbox"/>	czem5v74spc00087jmj0	✔ Completed	21 Mar 2025	21 Mar 2025	57s	Job	ibm_sherbrooke

Figura 34: Historial de ejecuciones de la tercera prueba

5.4.3. Tercera versión

Con el objetivo de mejorar los resultados obtenidos en las versiones anteriores, en esta tercera versión se decidió cambiar la formulación del problema. En lugar de procesar cada token individualmente y reemplazarlo de forma aislada, se cambia el planteamiento para predecir espacios marcados con un guion bajo (“_”) dentro de una frase. La idea es que estos espacios deben ser completados utilizando el contexto de palabras circundantes.

Este nuevo enfoque permite al circuito cuántico explotar el contexto alrededor de las palabras faltantes, procurando que las palabras generadas sean más coherentes con la oración.

El flujo de trabajo del circuito cuántico contextual es el siguiente:

- En primer lugar, al igual que en versiones anteriores, se entrena un modelo Word2Vec sobre un corpus en español para obtener embeddings vectoriales de 12 dimensiones que representan semánticamente cada palabra. Este corpus es reutilizado de una práctica de web scraping realizada en la universidad, y contiene noticias climatológicas.
- Se sustituyen palabras de las frases por huecos “_”. Para cada hueco, se calcula un vector de contexto promediando los embeddings de las dos palabras anteriores y las dos posteriores inmediatas al hueco, siempre que estas palabras existan en el vocabulario entrenado con Word2Vec. Si no existen palabras suficientes en el contexto, se completa con un vector de ceros.
- Este vector promedio del contexto se introduce al circuito cuántico. El circuito utilizado es similar a las versiones anteriores, consistiendo en puertas RX, RY y RZ para codificación, seguidas por una QFT, su inversa y finalmente un difusor de Grover. Sin embargo, en esta versión se eliminan las puertas CRZ y CNOT iniciales, simplificando la estructura y delegando al bloque QFT-Grover toda la generación de entrelazamiento global.
- Después de ejecutar el circuito en un simulador state-vector de quiskit sin ruido, se obtienen los valores esperados $\langle Z \rangle$ para cada qubit.
- Estos valores cuánticos se combinan linealmente al 50% con el embedding promedio del contexto. El vector resultante es utilizado para seleccionar la palabra más similar disponible en el vocabulario del modelo Word2Vec mediante la función *similar_by_vector*.
- Finalmente, la frase original se reconstruye insertando las palabras generadas en los huecos.

A continuación, se muestran ejemplos de resultados obtenidos:

```

Frase incompleta: El fuerte _ de lluvias que _ España en estos momentos
Texto generado : el fuerte metros de lluvias que vertientes españa en estos momentos
-----
Frase incompleta: Tras la _ de Valencia
Texto generado : tras la otras de valencia
-----
Frase incompleta: La _ martinho deja fuertes _ a su paso
Texto generado : la atlántica martinho deja fuertes cierta a su paso
-----
Frase incompleta: fuertes rachas de _
Texto generado : fuertes rachas de vientos
-----
Frase incompleta: la guardia _ activa la alerta _
Texto generado : la guardia civil activa la alerta responsables
-----

```

Figura 35: Enter Caption

Estos resultados muestran mejoras significativas con respecto a las versiones anteriores en términos de coherencia y variedad léxica. Por ejemplo, la frase “fuertes rachas de vientos” es semánticamente apropiada gracias al contexto utilizado.

Sin embargo, el circuito cuántico no tiene parámetros entrenables, lo que impide su optimización durante el entrenamiento. Al no introducir información adicional y actuar únicamente como una transformación del embedding generado por Word2Vec, no está claro que mejore la representación semántica inicial.

5.5. Quantum Long Short-Term Memory

La idea de este apartado es probar un modelo híbrido que integra redes LSTM con una capa cuántica variacional encargada de procesar parte de la información secuencial, y comparar el modelo con un LSTM clásico.

5.5.1. Conjunto de datos y preprocesamiento

Para este experimento se utiliza un dataset sintético que parte de un vocabulario de 11 palabras. A partir de este vocabulario se crearon las siguientes frases: ‘la computación cuántica procesa información’, ‘el modelo entrena nlp’, ‘nlp procesa el texto’, ‘la información entrena nlp’, ‘el texto procesa modelo’, ‘la computación nlp modelo’, ‘el modelo procesa información’, ‘la información texto procesa’, ‘computación entrena nlp modelo’ y ‘el texto entrena información’.

Para entrenar el modelo, a cada frase completa se le extrajo la última palabra, utilizando las tres anteriores como entrada y la cuarta como objetivo a predecir. Las oraciones con mayor longitud a 4 palabras se dividieron en 2 oraciones distintas.

La codificación de las oraciones se realiza mediante one-hot encoding. Esto significa que para un vocabulario de N palabras, cada palabra se convierte en un vector de dimensión N con todos los valores a cero salvo en la posición del índice de la palabra.

5.5.2. LSTM clásico

El modelo clásico utilizado es una red neuronal recurrente tipo LSTM diseñada específicamente para procesar secuencias de datos.

El objetivo del modelo en esta primera versión es predecir una palabra en base a una secuencia de tres palabras que recibe como entrada.

La LSTM está compuesta por celdas que mantienen una memoria interna, lo que les permite conservar información relevante de pasos anteriores en la secuencia. Incorpora 3 compuertas principales, la puerta de olvido, que decide qué información del estado anterior se descarta, la puerta de entrada, que decide qué información nueva debe almacenarse y la puerta de salida, que determina qué parte del estado interno se utilizará como salida en el paso actual.

Estas compuertas están controladas por funciones de activación sigmoid y tanh, lo que permite una gestión flexible y no lineal del flujo de información.

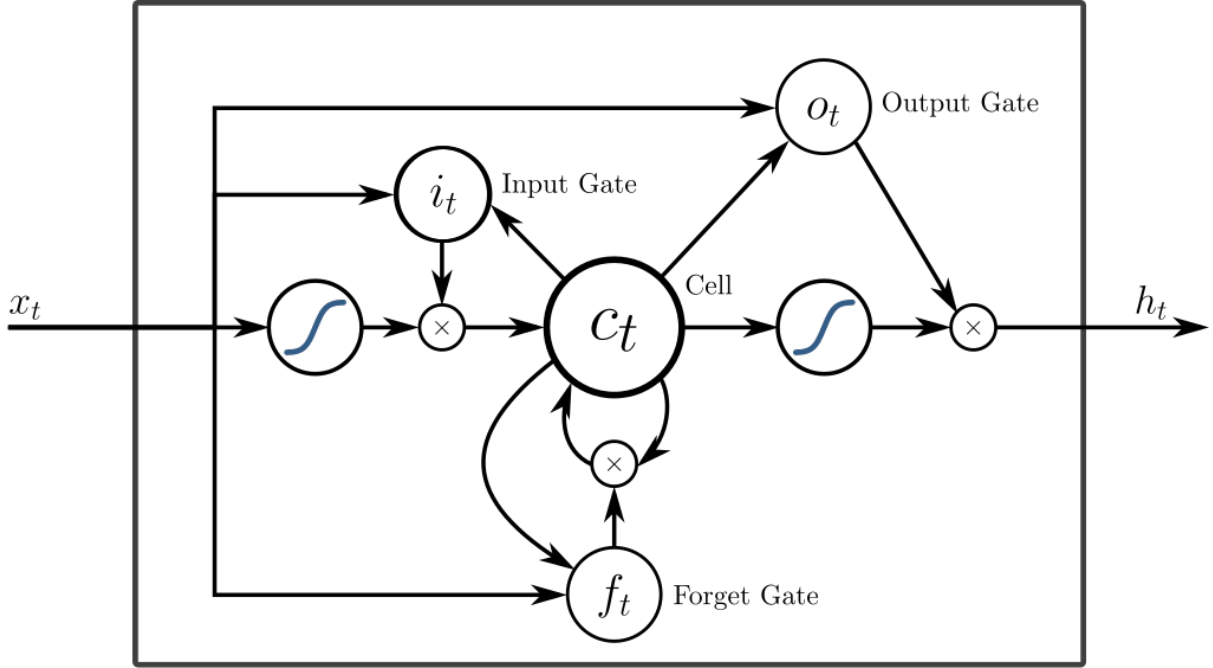


Figura 36: Arquitectura interna de una celda LSTM.

La salida de la LSTM, es un vector oculto que representa el contexto acumulado, y es procesado por un decoder lineal que proyecta la salida a una dimensión igual al tamaño del vocabulario. El modelo devuelve un vector de puntuaciones que se interpreta como una distribución de probabilidades sobre las posibles palabras a predecir, donde la palabra predicha es aquella que obtiene la puntuación más alta.

5.5.3. QLSTM híbrido

El modelo híbrido desarrollado se basa en la LSTM clásica, pero incorpora una capa cuántica variacional dentro de la celda recurrente. Mediante esta capa cuántica se busca aprovechar las propiedades de entrelazamiento y superposición para enriquecer el procesamiento de secuencias. La motivación de probar esta arquitectura para tareas de NLP generativo viene del trabajo propuesto por [46], donde integran circuitos cuánticos variacionales dentro de celdas LSTM clásicas para tareas de modelado secuencial. En este trabajo, demuestran que este enfoque puede aprender datos temporales de manera efectiva, y en algunos casos, logra converger más rápido o alcanzar una mejor precisión que la contraparte clásica.

A diferencia del modelo clásico, el modelo híbrido reemplaza el procesamiento interno de la celda LSTM por el siguiente circuito cuántico parametrizado:

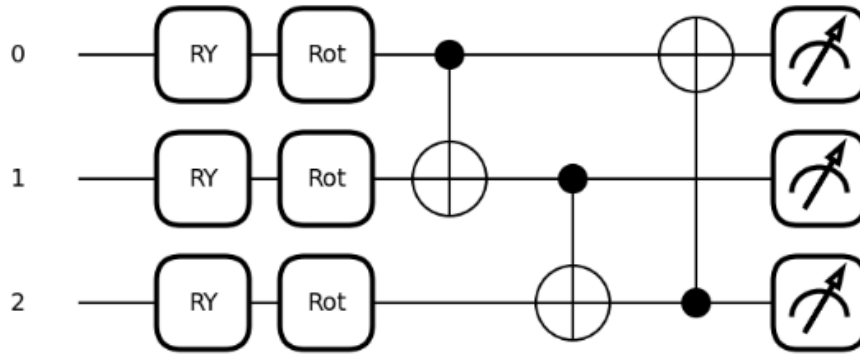


Figura 37: Circuito cuántico usado como capa dentro del modelo híbrido.

Este modelo es híbrido porque se sigue haciendo de forma clásica, el preprocesamiento de la entrada, que convierte las palabras en vectores one-hot, la concatenación de la entrada actual y el estado oculto previo, la proyección lineal hacia un número de qubits n , a través de una capa totalmente conectada, y el decoder final que convierte la salida cuántica en un vector de logits sobre el vocabulario.

La parte cuántica del modelo es el circuito presentado en la Figura 37. Este circuito consta de 3 qubits, suficiente para el corpus acotado y sintético utilizado durante el experimento. Al comienzo del circuito, cada qubit recibe una rotación sobre el eje Y. El ángulo de esta rotación se calcula a partir de una transformación lineal del input clásico, concatenando el estado oculto con la entrada actual.

Después de las rotaciones de entrada, se aplica una capa de compuertas Rot. Cada Rot, representa una combinación de RZ, RY y RX, que son los parámetros entrenables del circuito.

Posteriormente, los qubits se entrelazan utilizando puertas CNOT en patrón circular, permitiendo así que la información fluya entre todos los qubits del circuito y generando correlaciones no locales entre las partes del input. Finalmente, se mide el valor esperado del observable Z en cada qubit y se proyectan las mediciones al espacio clásico mediante una capa lineal para actualizar el estado oculto del modelo.

5.5.4. Entrenamiento y Resultados

Ambos modelos fueron entrenados sobre el mismo dataset durante 100 épocas utilizando el optimizador Adam, con una tasa de aprendizaje de 0.01 y la función de pérdida CrossEntropy. En cada época el modelo realiza un paso de optimización por ejemplo, ya que se entrena con batch size 1. No se aplicó validación cruzada debido al tamaño reducido del corpus, pero se realizó una evaluación sobre un subconjunto de los datos de entrenamiento para comparar el rendimiento final de los modelos dentro del mismo dataset utilizado.

Se registró la pérdida total por época para observar la dinámica de convergencia de ambos modelos:

En la figura 38 se observa que el modelo clásico converge ligeramente más rápido, mientras que el modelo híbrido tiene oscilaciones más prolongadas antes de estabilizarse. Aún así, el modelo híbrido logra alcanzar el rendimiento del modelo clásico.

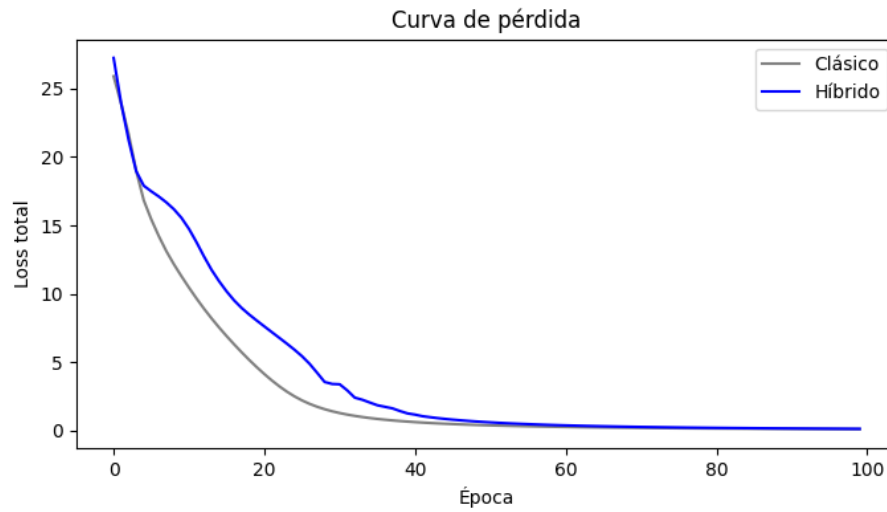


Figura 38: Gráfica de la curva de pérdida de la LSTM y la QLSTM

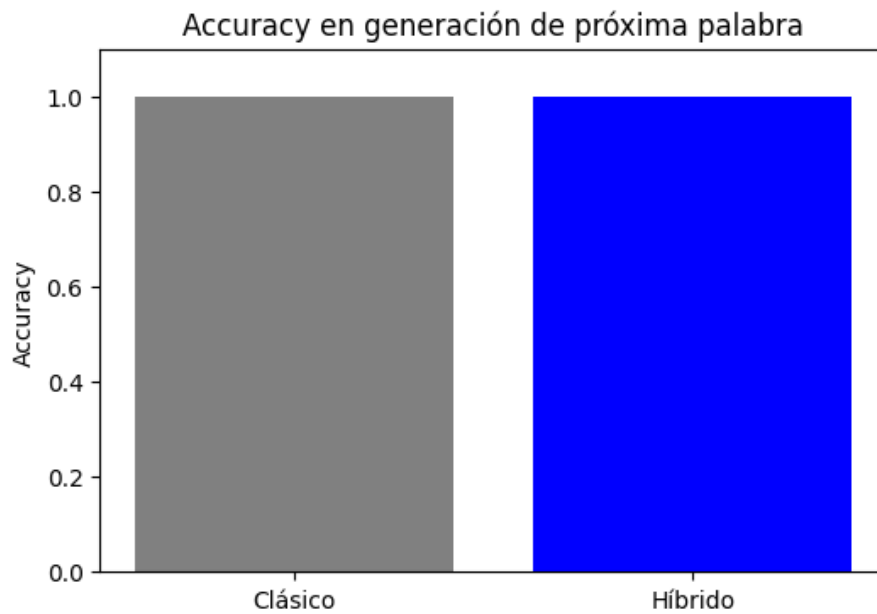


Figura 39: Accuracy alcanzada por los modelos

En la figura 39 se observa que ambos modelos consiguen un 100 % de acierto en el conjunto de test utilizado, lo cual sugiere que ambos han logrado memorizar correctamente el pequeño dataset.

En este experimento, no se observa una ventaja cuántica clara en términos de precisión o generalización, aunque en términos de complejidad, el modelo híbrido incorpora una capa cuántica con muy pocos parámetros, en este caso, 9 parámetros entrenables en la parte cuántica, lo que representa una reducción significativa en comparación con la LSTM puramente clásica.

Como se observa en la tabla 11, el modelo híbrido reduce el número de parámetros en un 74 %, a cambio de introducir una capa cuántica entrenable, lo que puede tener implicaciones futuras en eficiencia y capacidad representacional.

Componente	LSTM	QLSTM
Procesamiento secuencial	672	60
Capa cuántica (Rot gates)	0	9
Salida del estado oculto	0	32
Decoder final	99	99
TOTAL	771	200

Tabla 11: Resumen del número de parámetros por componente de los modelos.

5.5.5. Búsqueda de arquitectura cuántica

Con el fin de determinar la configuración cuántica más eficiente, se realizó un grid search sobre el número de qubits y la profundidad del circuito variacional. Cada combinación se entrenó hasta que el valor de pérdida cayó por debajo de 0.15 o se alcanzaron 100 épocas. La mejor configuración de la búsqueda fue de 5 qubits con 2 capas de profundidad, que redujo el número de épocas a la convergencia en un 25 % con respecto a la media.

```

===== Resultados ordenados por épocas hasta convergencia =====
q=5,d=2    + 73 épocas, val_loss=0.1482, tiempo=113.13s
q=5,d=3    + 81 épocas, val_loss=0.1484, tiempo=169.72s
q=4,d=2    + 96 épocas, val_loss=0.1496, tiempo=121.36s
q=4,d=3    + 99 épocas, val_loss=0.1495, tiempo=161.05s
q=3,d=1    + 100 épocas, val_loss=0.4019, tiempo=74.21s
q=3,d=2    + 100 épocas, val_loss=0.454, tiempo=93.51s
q=3,d=3    + 100 épocas, val_loss=0.186, tiempo=121.52s
q=4,d=1    + 100 épocas, val_loss=0.1847, tiempo=81.8s
q=5,d=1    + 100 épocas, val_loss=0.9655, tiempo=104.27s

```

Figura 40: Resultados del Grid Search para la QLSTM

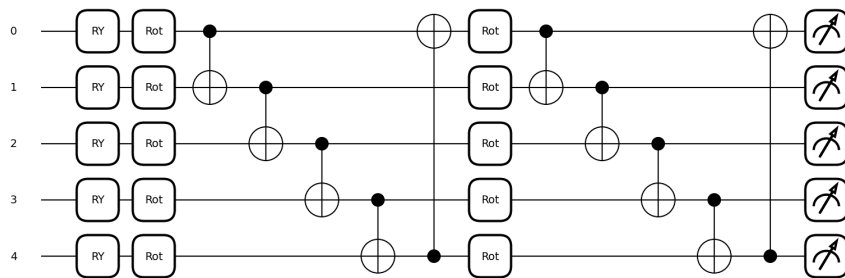


Figura 41: Circuito cuántico usado como capa dentro del modelo híbrido con 5 qubits y 2 capas de profundidad

El código se volvió a ejecutar pero utilizando el circuito de la Figura 41 en lugar del circuito de la Figura 37. El accuracy volvió a ser del 100 % pero hubo ligeros cambios en la convergencia del modelo.

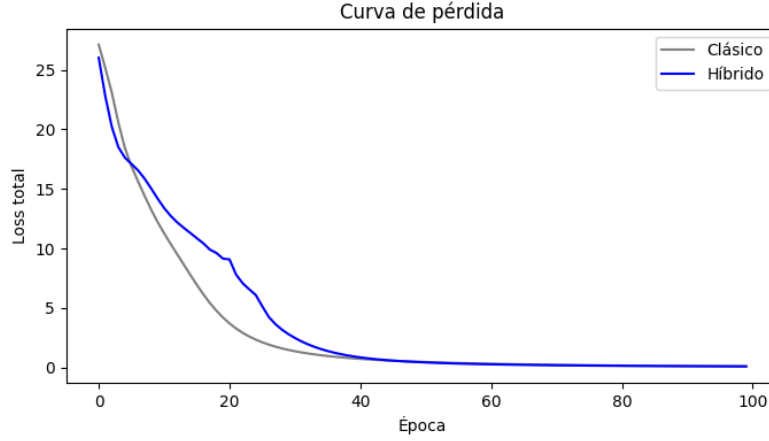


Figura 42: Gráfica de la curva de pérdida de la LSTM y la QLSTM con 5 qubits y 2 capas de profundidad

En la figura 42 se observa que, a pesar de la diferencia con la versión anterior, la versión clásica sigue convergiendo ligeramente más rápido. Además, al añadir qubits y profundidad al circuito, también se aumenta el número de parámetros en la versión híbrida.

5.5.6. QLSTM con cuatro puertas cuánticas y re-uploading

Hasta ahora, el modelo híbrido incorporaba una única capa cuántica variacional en sustitución directa del procesamiento interno de una celda LSTM clásica. Esta capa recibía una combinación del estado oculto anterior y el input actual, y tras pasar por el circuito cuántico devolvía un vector que actuaba directamente como el nuevo estado oculto.

En esta nueva versión se introducen varias mejoras relevantes respecto a la arquitectura inicial. El primero de los cambios es que, en lugar de utilizar un único circuito, ahora cada QLSTM contiene cuatro compuertas cuánticas independientes que replican exactamente la lógica funcional de una LSTM clásica:

- Puerta de entrada (i_t)
- Puerta de olvido (f_t)
- Puerta de salida (o_t)
- Puerta de actualización del estado interno (g_t o \tilde{c}_t)

En esta versión, cada una de estas compuertas es un circuito cuántico independiente con sus propios parámetros entrenables, procesando la misma concatenación de la entrada actual y el estado oculto anterior. La salida de estos circuitos se combina posteriormente según las ecuaciones estándar de la LSTM:

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t, \quad h_t = o_t \odot \tanh(c_t)$$

Los cuatro circuitos cuánticos que actúan como puertas en la QLSTM tienen exactamente la misma estructura, idéntica a la de la Figura 41. Aunque la arquitectura es idéntica, cada puerta mantiene su propio tensor de pesos, por lo que aprende transformaciones diferentes adaptadas a su función. Esto permite que la celda híbrida tenga una correspondencia directa con la estructura de la LSTM original, favoreciendo una mejor integración entre los componentes clásicos y cuánticos.

Otra novedad introducida es el uso de la técnica *data re-uploading* [47]. Normalmente, un circuito cuántico recibe la información al principio, realiza operaciones cuánticas, y finalmente produce un resultado. En cambio, con esta técnica, la información se “carga” varias veces intercalada con operaciones cuánticas adicionales, en este caso, rotaciones y entrelazamientos. Lo que se pretende conseguir con esta técnica es que, con el mismo número de qubits y profundidad física, se incremente la capacidad expresiva del circuito, ya que puede capturar representaciones más ricas y complejas mediante reiteradas exposiciones del input clásico. En esta versión, el número de re-uploads está fijado en 2.

Debido al aumento de parámetros cuánticos en esta versión, el entrenamiento se vuelve más inestable, por lo que usan otras técnicas como clipping del gradiente, dropout del 10 % antes del decodificador final y optimizador AdamW con una tasa de aprendizaje de 0.005.

Después del entrenamiento, esta versión mejorada de la QLSTM presenta curvas de pérdida más estables y cercanas a la versión puramente clásica, sin grandes fluctuaciones iniciales, y manteniendo el accuracy del 100 % sobre el dataset sintético.

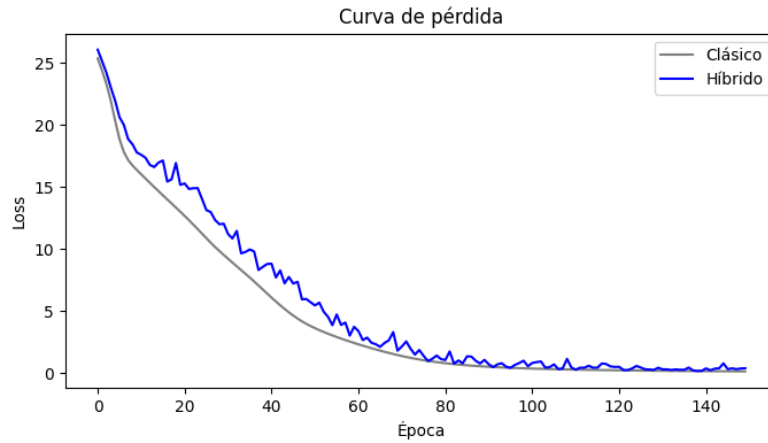


Figura 43: Gráfica de la curva de pérdida de la LSTM y la QLSTM con 4 circuitos cuánticos

Además, el re-uploading ofrece una mayor flexibilidad para escalar el modelo hacia vocabularios más amplios o secuencias más largas sin necesidad inmediata de aumentar la cantidad de qubits físicos, algo especialmente relevante teniendo en cuenta la limitación actual del hardware cuántico.

Debido al interesante resultado obtenido en esta versión, se decidió hacer la comparativa pero con un dataset ampliado. El dataset empleado en esta ocasión se compone de 50 frases sintéticas generadas a partir de un vocabulario de 15 palabras agrupadas en categorías:

Categoría	Elementos
det	el, la
noun1	computación, modelo, sistema
noun2	cuántica, clásico, nlp
verb	procesa, entrena, analiza, clasifica
objeto	información, texto, datos

Tabla 12: Categorías léxicas usadas para la construcción del dataset.

Cada frase se construye de forma aleatoria siguiendo la estructura fija: determinante + sustantivo1 + sustantivo2 + verbo + objeto, generando combinaciones como “la computación cuántica analiza datos”. A partir de estas frases se extraen ejemplos de entrenamiento tomando secuencias de tres palabras consecutivas como contexto y la siguiente palabra como objetivo a predecir. De esta forma, se obtienen 200

pares que se dividen aleatoriamente en entrenamiento y validación.

Al entrenar tanto la LSTM como la QLSTM de la misma forma y con la misma estructura explicada anteriormente, cambiando solo el dataset, se obtienen los siguientes resultados:

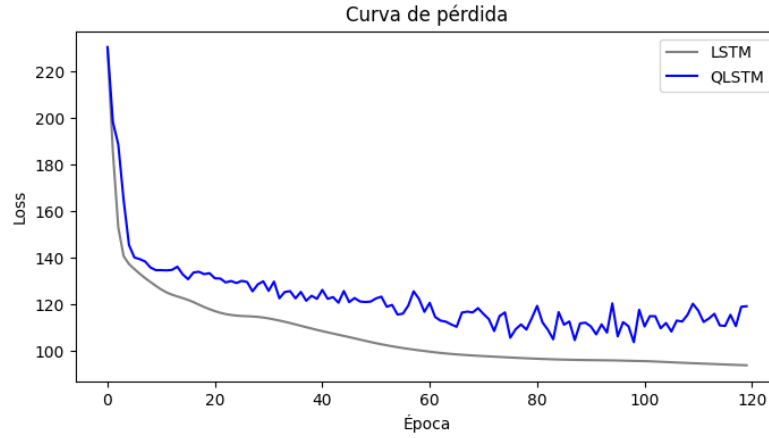


Figura 44: Gráfica de la curva de pérdida para dataset con 50 frases.

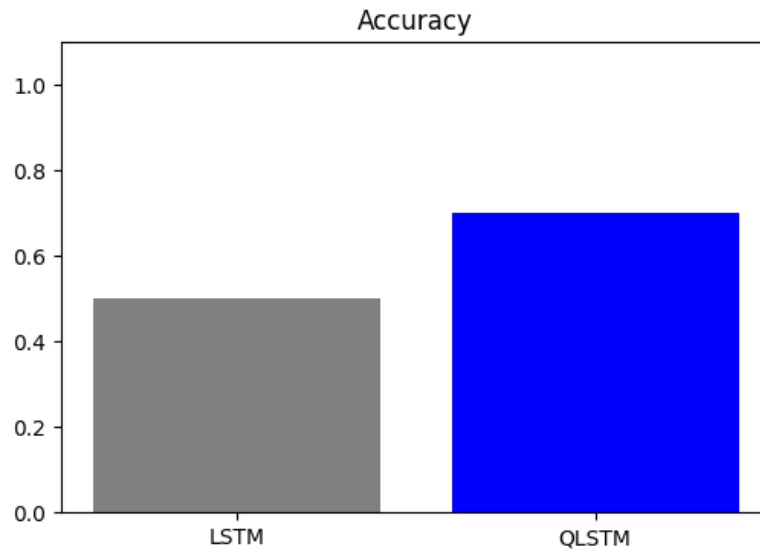


Figura 45: Accuracy para dataset con 50 frases.

En la Figura 45 se muestra que el modelo híbrido alcanza valores de precisión superiores a los de la LSTM clásica. No obstante, debido al tamaño reducido del conjunto de validación (10 ejemplos), cada predicción correcta implica una variación en la precisión de $\pm 10\%$, dificultando extraer conclusiones definitivas basadas únicamente en esta configuración experimental.

En cuanto al comportamiento durante el entrenamiento, en la Figura 44 se observa que ambos modelos reducen progresivamente su pérdida, aunque la curva de aprendizaje del modelo híbrido muestra ligeras fluctuaciones, debido a la complejidad adicional de las operaciones cuánticas. Además, el coste computacional del entrenamiento aumenta debido al uso del simulador cuántico, siendo este aproximadamente 3 a 4 veces superior al del modelo clásico.

Se decidió repetir la ejecución con otras 5 semillas diferentes a la anterior, manteniendo el conjunto de datos para comprobar si la QLSTM era sistemáticamente superior a la LSTM o si había sido una mera coincidencia. Los resultados de las ejecuciones se muestran en la siguiente tabla:

Seed	LSTM	QLSTM
42	0.40	0.40
1337	0.40	0.40
2025	0.40	0.50
31415	0.40	0.20
777	0.40	0.60
Media \pm Desv.	0.400 ± 0.000	0.420 ± 0.133

Tabla 13: Comparativa de accuracy entre LSTM y QLSTM en 5 repeticiones con distintas semillas.

Como se observa en la Tabla 13, la LSTM ofrece un accuracy constante de 0.40, mientras que la QLSTM oscila con valores entre 0.20 y 0.60. Aunque es cierto que la QLSTM promedia un accuracy más alto, la inestabilidad de sus precisiones muestra que es más sensible a la inicialización y puede atascarse en óptimos locales. Por lo tanto, aún no se observa una ventaja cuántica robusta en términos de precisión. Aún así, los resultados son bastante interesantes y muestran que merece la pena seguir investigando esta arquitectura híbrida.

En términos de parámetros, en esta versión, debido a aumentar el número de qubits y profundidad del circuito, sumado a que se usan 4 circuitos, y teniendo en cuenta el reupload, el número de parámetros del modelo híbrido asciende considerablemente, llegando a superar al modelo clásico:

Componente	LSTM Generator	QLSTM Gated Generator
Pre-procesamiento secuencial	672	400
Capa cuántica	0	240
Salida del estado oculto	0	192
Decoder final	99	99
TOTAL	771	931

Tabla 14: Comparativa de parámetros entre el modelo LSTM Generator y el modelo QLSTM Gated Generator.

En esta ocasión, el modelo híbrido supera al clásico en número de parámetros en un 21 %. No obstante, el grid search para determinar el número de qubits y profundidad se realizó sobre un modelo híbrido con un único circuito cuántico por motivos de tiempo de ejecución. Sería interesante investigar si al utilizar 4 circuitos dentro del mismo modelo, 5 qubits y 2 capas de profundidad sigue siendo la mejor combinación.

6. Valoración ética del proyecto

Este capítulo integra la reflexión ética del proyecto. La redacción se basa en los recursos para PFM que la universidad pone a disposición del alumnado.

6.1. Metodología de análisis ético

En primer lugar, se identifican las cuestiones morales que atraviesan este trabajo. Después, se aplican los principios de la ética de la tecnología: respeto a la autonomía, beneficencia, no maleficencia y justicia, y se contrastan con los Objetivos de Desarrollo Sostenible y con la normativa vigente sobre protección de datos. La combinación de un enfoque deontológico y uno consecuencialista permite realizar un análisis equilibrado y accionable.

6.2. Cuestiones éticas relevantes

El proyecto se sitúa en la convergencia, todavía poco transitada, entre inteligencia artificial y computación cuántica. Esta convergencia plantea retos específicos, entre los que destacan tres:

- **Privacidad de los datos lingüísticos:** Incluso los corpus abiertos pueden contener fragmentos que identifiquen indirectamente a personas.
- **Reproducción de sesgo:** Si el modelo aprende prejuicios de género o raza, terminará amplificándolos. Sin un filtro crítico, los modelos pueden convertirse en un altavoz de estereotipos.
- **Impacto Ambiental:** Si bien los simuladores clásicos consumen recursos computacionales significativos, los ordenadores cuánticos actuales requieren un gasto energético aún mayor para mantener temperaturas criogénicas cercanas al cero absoluto.

6.3. Evaluación de impactos y medidas de mitigación

Desde la perspectiva socio-económica, la investigación abre oportunidades para arquitecturas de IA más eficientes que, a largo plazo, podrían reducir la barrera de entrada a la generación automática de texto multilingüe. Por otro lado, también existe la posibilidad de que la brecha digital se ensanche entre aquellas organizaciones que puedan permitirse acceder a hardware cuántico y aquellas que siguen atadas a infraestructuras tradicionales. Se recomienda fomentar la publicación abierta de herramientas y resultados para evitar barreras de entrada.

En lo cultural, crear lenguaje de forma automática trae consigo la responsabilidad de proteger la diversidad de dialectos, para que aquellos menos visibles no se queden sin voz. Para mitigar el riesgo de homogeneización cultural, se propone incluir corpus variados y trabajar en colaboración con expertos en lingüística y comunidades locales.

En cuanto al impacto medioambiental, la huella actual de las tecnologías cuánticas se concentra en la intensa demanda energética de la refrigeración criogénica, que puede consumir hasta 25 kW de potencia continua por sistema [48], del uso de los isótopos helio-3/4, cada vez más escasos y costosos [49], y de la extracción de metales estratégicos como el niobio, cuya minería conlleva deforestación y pérdida de biodiversidad. En futuras fases, se recomienda monitorizar el impacto ambiental y explorar alternativas más sostenibles.

En cuanto a los corpus, se utilizaron mayoritariamente corpus creados manualmente para experimentos pequeños, siempre neutrales en ideologías y sin ningún tipo de datos que pudiesen vulnerabilizar la privacidad de cualquier persona.

6.4. Conclusiones

Al igual que con las tecnologías clásicas, el desarrollo de tecnologías de procesamiento del lenguaje natural basadas en computación cuántica requiere un enfoque ético transversal y dinámico. La identificación temprana de riesgos permite anticipar y mitigar potenciales efectos adversos. No obstante, dado el carácter emergente y disruptivo de la tecnología cuántica, se considera fundamental adoptar una actitud de vigilancia continua, actualización normativa y apertura a la participación de la sociedad civil y los grupos afectados. Sólo así será posible promover un avance científico que respete los derechos fundamentales, fomente la equidad y contribuya a la sostenibilidad global.

7. Plan de trabajo

El desarrollo del proyecto se organizó en distintas fases, asegurando una adecuada gestión del tiempo y los recursos disponibles, así como la consecución de objetivos planteados de forma eficaz.

7.1. Fases

Fase 1: Analizar el estado del arte de Quantum Natural Language Processing

- Revisión bibliográfica sobre procesamiento de lenguaje natural y computación cuántica.
- Establecimiento de los objetivos y alcance del proyecto.

Fase 2: Desarrollo experimental

- Implementación y validación de modelos cuánticos para NLP.
- Ejecución de experimentos en simuladores y, cuando sea viable y se espere cierta utilidad, en hardware real.
- Documentación sistemática y recopilación de resultados.

Fase 3: Análisis y discusión de resultados

- Análisis crítico de los resultados obtenidos.
- Comparación de rendimiento de modelos cuánticos frente a métodos clásicos.

Fase 4: Valoración ética y social

- Análisis de aspectos éticos, sociales y medioambientales.
- Identificación y propuestas de mitigación de riesgos potenciales.

Fase 5: Finalización y documentación

- Redacción final del documento del proyecto.
- Revisión, ajustes y preparación para la defensa académica del trabajo.

7.2. Tareas

A continuación se presenta una tabla con las tareas a realizar en cada fase:

Fase	Tarea	Descripción
1	1.1	Revisar conocimientos en NLP
	1.2	Estudiar fundamentos de la computación cuántica
	1.3	Leer artículos sobre investigaciones y proyectos en Quantum NLP y quantum generative
	1.4	Definir objetivos y alcance del proyecto
	1.5	Documentar nuevos aprendizajes
2	2.1	Experimentación con QCBM
	2.2	Experimentación con Simulated Annealing y SWAP Test
	2.3	Experimentación con Transformer con auto-atención cuántica
	2.4	Experimentación con QLSTM
3	3.1	Documentación sistemática
	3.2	Comparación con modelos clásicos
	3.3	Análisis crítico de los resultados obtenidos.
4	4.1	Documentación de las conclusiones obtenidas
	4.2	Análisis de aspectos éticos, sociales y medioambientales. Identificación y propuestas de mitigación de riesgos potenciales.
5	5.1	Terminar de redactar la memoria
	5.2	Revisión y ajustes en el proyecto
	5.3	Preparar repositorio en GitHub
	5.4	Preparar defensa

Tabla 15: Listado de tareas del proyecto organizadas por fase

7.3. Diagrama de Gantt

A continuación se presenta el diagrama de Gantt creado para el Proyecto Fin de Máster.

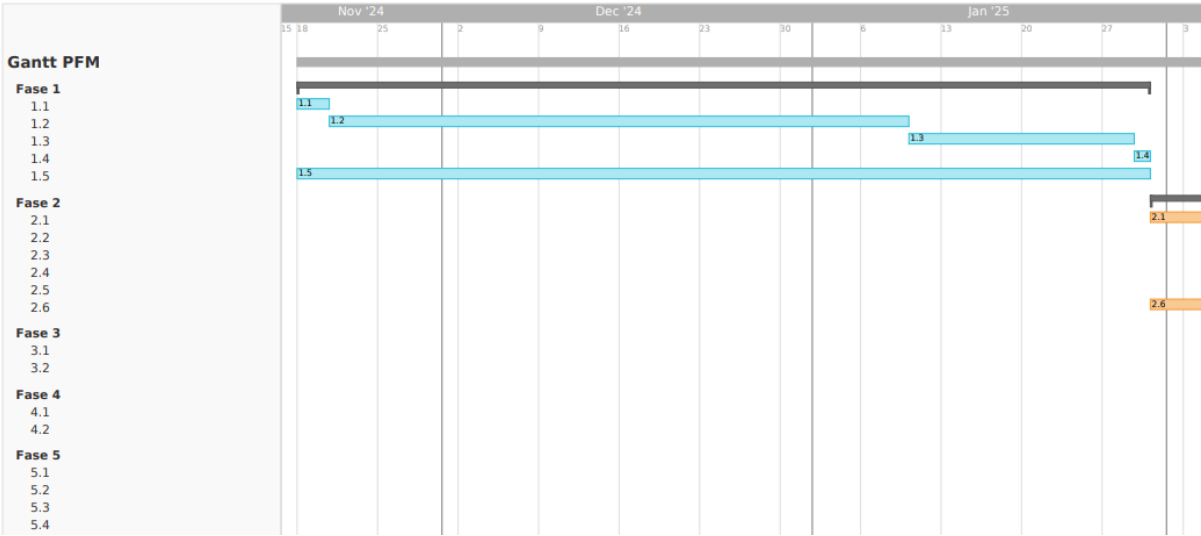


Figura 46: Diagrama de Gantt parte 1

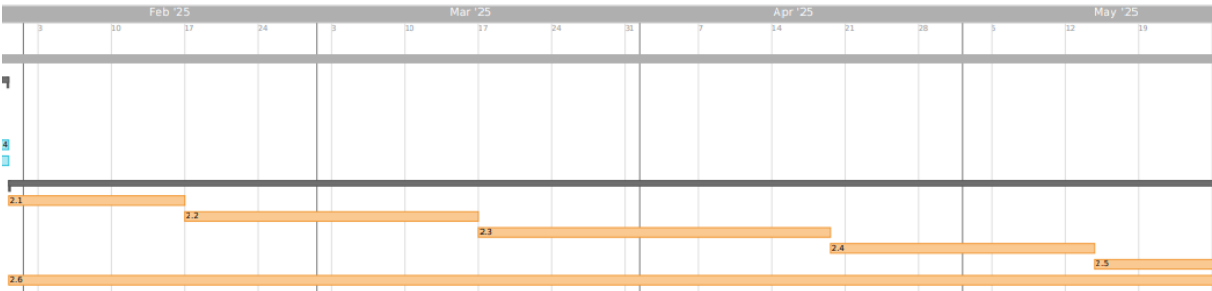


Figura 47: Diagrama de Gantt parte 2

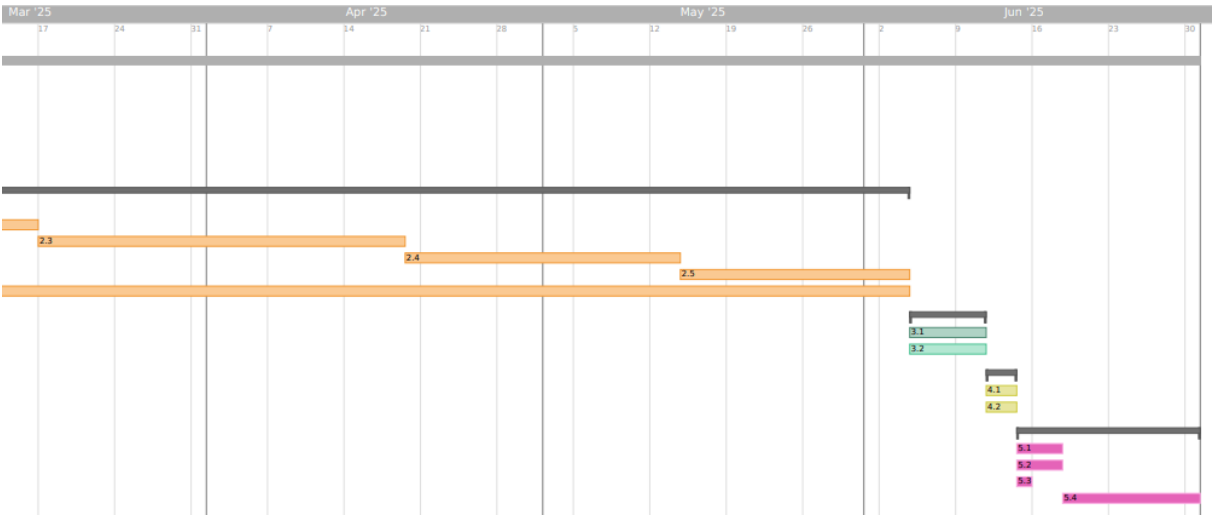


Figura 48: Diagrama de Gantt parte 3

8. Presupuesto

El objetivo de este apartado es proporcionar una estimación detallada y transparente de los costes asociados a este proyecto. El presupuesto se ha calculado teniendo en cuenta el tiempo invertido por cada profesional, el coste de los materiales utilizados, y otros gastos asociados directamente al desarrollo de este proyecto.

Este análisis detallado de costes proporciona una visión clara y completa del presupuesto necesario para llevar a cabo el proyecto, facilitando una comprensión profunda del alcance económico del mismo.

En la tabla 16 se muestra el presupuesto del proyecto relacionado con la parte de recursos humanos.

Perfil	Horas	Precio	Total
Director PFG	15	50€/h	750€
Responsable Investigador	10	30€/h	300€
Estudiante Investigador	450	13€/h	5850€
Redactor técnico	60	10€/h	600€

Tabla 16: Resumen de horas y costes por perfil profesional.

En la tabla 17 se muestra el presupuesto del proyecto relacionado con la parte material.

Material	Precio	Total
Desplazamientos	0,06€/km	266,34€
Ordenador portátil	600€/ud	600€
Periféricos	90€/ud	90€
Windows 11	145€/ud	145€
Google Colab Pro+	51,12€/mes	408,96€
IBM Quantum	0€	0€
Overleaf	0€	0€

Tabla 17: Costes asociados al material y software utilizado.

A continuación se presenta una tabla resumen de gastos

Tipo	Coste
Recursos humanos	7500€
Recursos materiales	1519,3€
Total	9019,3€

Tabla 18: Resumen del presupuesto del proyecto

Este desglose económico no solo permite cuantificar los recursos necesarios, sino que también respalda la viabilidad del proyecto al mostrar una distribución equilibrada entre costes humanos y materiales.

9. Estructura del proyecto

9.1. Estructura del repositorio

Todo el código fuente desarrollado durante este Proyecto Fin de Máster ha sido subido y organizado en un repositorio público de GitHub bajo el nombre **PFM-Iker-Marcelo**, disponible en la rama principal.

El repositorio contiene dos carpetas principales:

- **código:** Contiene los notebooks implementados para los distintos experimentos del proyecto. Está estructurada en subcarpetas según el enfoque o modelo evaluado:
 - **QCBM:** experimentos de generación de texto con Quantum Circuit Born Machines.
 - **QGAN:** implementación de la arquitectura híbrida Quantum GAN.
 - **QLSTM:** red neuronal híbrida que integra una capa cuántica dentro de una LSTM clásica.
 - **QSAT:** auto-atención cuántica basada en circuitos inspirados en QFT y Grover.
 - **SWAP_TEST:** generadores basados en búsqueda clásica guiada por SWAP test.
- **documentación:** carpeta destinada a almacenar documentación adicional, aún en desarrollo en el momento de escribir esta memoria.

A través de este repositorio es posible consultar, reproducir y modificar todos los experimentos descritos a lo largo del documento. Todos los códigos han sido ejecutados en Google Colab, y para algunos de ellos es necesario activar la configuración de alta capacidad de RAM.

Además, algunos de los notebooks contienen pruebas y experimentos no documentados en la presente memoria, ya sea por redundancia o por que se consideró que no aportaban demasiado valor en cuanto a conclusiones y entendimiento del funcionamiento de los enfoques.

El repositorio puede consultarse en [50].

10. Conclusiones y líneas futuras

10.1. Conclusiones

Este proyecto ha demostrado, a través de diversos prototipos, que es posible abordar tareas básicas de procesamiento y generación de lenguaje natural con computación cuántica. La memoria y experimentos realizados sirven como referencia pionera en castellano, facilitando que investigadores y estudiantes se introduzcan en este campo emergente, algo que a mí, antes de comenzar el proyecto me resultó bastante complicado. En términos prácticos, los resultados obtenidos evidencian un rendimiento competitivo pese a las limitaciones de hardware actuales. Estos hallazgos refuerzan el valor del trabajo tanto como prueba de concepto científica como por su aplicabilidad futura, ya que sientan las bases para que conforme madure la tecnología cuántica, se puedan escalar estas soluciones e incorporarlas en aplicaciones reales de procesamiento de lenguaje. Personalmente, considero que el aporte de este proyecto es doble, académicamente, enriquece el estado del arte con resultados y discusiones detalladas; y prácticamente, orienta sobre qué enfoques cuánticos son viables hoy y cómo podrían aprovecharse en el futuro inmediato.

Un aspecto destacable es el rol divulgativo y formativo de este trabajo dentro de la comunidad hispanohablante. Al compilar teoría y experimentos de Quantum NLP en español, la memoria llena un vacío en la bibliografía, permitiendo que profesionales accedan al tema y lo utilicen como base para otros proyectos. Esto tiene un impacto académico evidente, pero también un impacto social y económico a medio plazo, ya que facilita la formación de talento local especializado en computación cuántica aplicada a IA, lo cual es clave para sumarse a la vanguardia tecnológica.

Además, los resultados y análisis presentados tienen implicaciones que trascienden el ámbito académico. En el plano industrial, apuntan a un futuro donde las empresas podrán aprovechar la computación cuántica para impulsar aplicaciones de NLP más potentes. Gigantes tecnológicos ya exploran esta sinergia, por ejemplo, IBM anunció prototipos de modelos cuánticos capaces de traducir múltiples idiomas simultáneamente, anticipando mejoras en eficiencia y precisión en traducción automática[51] [52]. Del mismo modo, Google y otras compañías investigan sistemas cuánticos de reconocimiento de voz multilingüe en paralelo [51] [32]. Estos indicios sugieren que algunos de los enfoques estudiados en este trabajo podrían en el futuro integrarse en pipelines productivos. El valor práctico de una QLSTM con menos parámetros, por ejemplo, implica modelos más ligeros que consumirían menos memoria y energía en despliegues reales, aspecto atractivo para la industria una vez que el hardware cuántico esté listo. Es importante resaltar que este trabajo, al evidenciar tales posibilidades, aporta información valiosa para la I+D empresarial, ya que sirve de orientación temprana sobre qué algoritmos cuánticos de NLP van mostrando mayor promesa y cuáles son sus requerimientos.

Una pregunta fundamental que aborda este estudio es cómo se comparan los enfoques cuánticos de NLP frente a las técnicas clásicas actuales, y en qué escenarios específicos la aproximación cuántica puede ser más eficiente o relevante. A partir de los experimentos realizados y de la literatura reciente, se pueden extraer varias conclusiones al respecto:

- Los modelos cuánticos pueden representar distribuciones o relaciones en espacios de estado de alta dimensión de forma compacta gracias a la superposición y el entrelazamiento. En este trabajo, por ejemplo, un circuito QCBM con solo 9 qubits logró aprender con precisión una distribución sobre 512 estados posibles. Un modelo clásico para capturar una distribución de similar complejidad requiere cientos de parámetros, al menos uno por estado, mientras que el modelo cuántico codifica de forma implícita esa información en un estado cuántico global. Este potencial de compresión cuántica sugiere que, a medida que dispongamos de más qubits estables, las técnicas cuánticas podrían manejar vocabularios extensos o múltiples atributos semánticos sin sufrir un crecimiento exponencial de recursos que como los métodos clásicos. Gracias a la superposición, donde los enfoques clásicos tendrían que iterar numerosas combinaciones, un circuito cuántico bien diseñado las procesa en paralelo, brindando ventajas teóricas de velocidad y capacidad de almacenamiento.
- Los algoritmos de machine learning clásicos suelen requerir grandes conjuntos de datos y recursos

para generalizar bien. Sin embargo, algunos experimentos indican que los modelos cuánticos híbridos pueden destacar cuando los datos son escasos pero la tarea involucra patrones complejos. En este TFM, el discriminador cuántico de la QGAN logró sobrepasar el 94 % de precisión entrenando con tan solo 60 frases. Por otro lado, estudios externos han demostrado que añadir una capa cuántica a un modelo de lenguaje puede mejorar la clasificación en dominios con datos reducidos o muy ruidosos, gracias a que la entrelazamiento cuántico permite detectar correlaciones globales que a los métodos clásicos les cuesta discernir [53].

- El lenguaje natural tiene estructuras jerárquicas y dependencias a largo plazo. En enfoques clásicos, manejar estas dependencias requiere arquitecturas complejas como transformers con muchos cabezales de atención, o memorias recurrentes de gran tamaño. Las técnicas cuánticas, en cambio, tratan la composición gramatical de forma nativa en ciertos modelos como DisCoCat, donde la estructura sintáctica se traduce directamente en un diagrama o circuito que combina estados cuánticos. Esto es relevante porque, teóricamente, un modelo cuántico puede representar estados combinados de palabras con menos recursos de los que necesitaría un modelo clásico para representar el tensor de esas mismas palabras. Aunque en este proyecto el enfoque DisCoCat manejó frases cortas con vocabulario filtrado, alcanzó una precisión cercana a la clásica, insinuando que conforme se superen problemas de escala, la fusión de significado mediante entrelazamiento podría competir o incluso mejorar la eficiencia de los mecanismos de atención clásicos.
- Es importante matizar que, si bien las ventajas mencionadas son prometedoras, aún son potenciales o limitadas a casos de estudio reducidos y toy-models. Hoy por hoy, los modelos clásicos dominan prácticamente todas las tareas de NLP en métricas absolutas. No obstante, el campo de QNLP permite imaginar la posibilidad de lograr una ventaja cuántica real en procesamiento lingüístico a largo plazo. Esto ocurrirá cuando un sistema cuántico logre resolver un problema de NLP más rápido o más preciso que cualquier sistema clásico razonable. Ya se ha argumentado que la computación cuántica tiene el potencial de superar a los modelos clásicos tanto en eficiencia como en exactitud en el manejo de estructuras lingüísticas [54], siempre que dispongamos del nivel suficiente de recursos cuánticos.

10.2. Líneas Futuras

Dado que la computación cuántica para NLP se encuentra todavía en una fase muy temprana, muchas de las líneas futuras se centran en superar las barreras técnicas identificadas. Una prioridad absoluta es avanzar en la escalabilidad del hardware cuántico y la mitigación de errores. Los dispositivos actuales son ruidosos y de tamaño limitado, lo que nos forzó en este proyecto a trabajar con toy models. Hacia el futuro, será imprescindible implementar técnicas de corrección de errores cuánticos y reducción de ruido que permitan ejecutar circuitos más profundos sin degradación de la señal. Esto incluye el desarrollo de qubits más estables, mejores calibraciones y algoritmos de error mitigation en tiempo real. A medida que estos avances ocurran, podremos aumentar el número de qubits utilizables en experimentos de NLP cuántico y, por ende, abordar oraciones más largas y vocabularios más extensos. Otro aspecto crítico es la disponibilidad de una memoria cuántica eficiente. Hoy por hoy no existe una RAM cuántica funcional a gran escala, lo que limita enormemente la capacidad de los algoritmos para almacenar y reutilizar información intermedia. Desarrollar arquitecturas de memoria cuántica sería revolucionario, por ejemplo, permitiría cargar grandes conjuntos de datos textuales en estados cuánticos para su procesamiento directo, algo inviable actualmente. Si bien la construcción de una QRAM es compleja, explorar alternativas intermedias será una vía de investigación. También deberán investigarse optimizadores cuánticos avanzados para el entrenamiento de modelos QNLP. Desarrollar métodos de optimización específicos para circuitos variacionales de NLP, quizás inspirados en optimizadores adaptativos clásicos como Adam, pero compatibles con la naturaleza probabilística de la medición cuántica, podría acortar los tiempos de entrenamiento y mejorar la convergencia de los modelos cuánticos.

Otro camino prometedor y de gran interés es la integración de algoritmos cuánticos con los grandes modelos de lenguaje clásicos que hoy lideran el campo del NLP. Sería interesante continuar la investigación de este Proyecto Fin de Máster con arquitecturas híbridas donde componentes cuánticos mejoren aspectos

específicos de los modelos clásicos. Este enfoque ya se trabaja en investigaciones recientes. por ejemplo, IonQ presentó en 2025 un método híbrido cuántico-clásico para fine-tuning de modelos tipo Transformer, añadiendo un circuito cuántico como cabeza de clasificación a un sentence transformer preentrenado [55]. En [55], los resultados iniciales mostraron que ciertas configuraciones cuánticas podían superar la precisión de enfoques puramente clásicos en tareas de clasificación de texto. A corto plazo, es factible que sigamos esta línea incorporando capas cuánticas dentro de arquitecturas transformadoras, aprovechando que la atención y otras operaciones son matemáticamente compatibles con operaciones cuánticas.

Finalmente, las líneas futuras deben dirigirse a expandir el alcance de Quantum NLP hacia nuevas aplicaciones y, críticamente, buscar demostrar de forma palpable las ventajas cuánticas en escenarios concretos. Hasta ahora, gran parte de los esfuerzos (incluido el presente trabajo) se han concentrado en validar que “es posible” aplicar modelos cuánticos a tareas sencillas de NLP. El siguiente paso natural será aumentar la complejidad de las tareas abordadas y acercarlas a problemas de la vida real. Ahora bien, para que estas aplicaciones destaquen, es fundamental perseguir de manera explícita la ventaja cuántica. Esto significa diseñar experimentos donde esperemos que lo cuántico supere claramente a lo clásico. Una estrategia será identificar tareas de NLP que se alineen con algoritmos cuánticos de velocidad superior.

Adicionalmente, será importante seguir explorando modelos híbridos cuántico-clásicos pero llevando la balanza cada vez más hacia la parte cuántica conforme esta sea capaz. En un horizonte de 5-10 años, podríamos pasar de pequeños componentes cuánticos en flujos mayormente clásicos, a que los modelos cuánticos asuman la porción principal de la tarea, apoyados aún por pre/post-procesamiento clásico.

Referencias

- [1] Kevin Kherb. Bloch sphere visualizer, 2024. <https://bloch.kherb.io/>.
- [2] Wikipedia contributors. Superposición cuántica — wikipedia, la enciclopedia libre, 2024. https://es.wikipedia.org/wiki/Superposici%C3%B3n_cu%C3%A1ntica.
- [3] IBM Quantum. Ibm quantum composer, 2024. <https://quantum.ibm.com/composer>.
- [4] Aleks Kissinger. An introduction to quantum natural language processing, 2021. <https://medium.com/qiskit/an-introduction-to-quantum-natural-language-processing-7aa4cc73c674>.
- [5] IBM Quantum. Quantum workloads — ibm quantum, 2025. <https://quantum.ibm.com/workloads>.
- [6] Wikipedia colaboradores. Procesamiento de lenguajes naturales — wikipedia, la enciclopedia libre, 2025. https://es.wikipedia.org/wiki/Procesamiento_de_lenguajes_naturales.
- [7] IBM. ¿qué es el procesamiento del lenguaje natural? — ibm, 2025. <https://www.ibm.com/mx-es/think/topics/natural-language-processing>.
- [8] Andrés Montoro-Montarroso, Javier Cantón-Correa, Paolo Rosso, Berta Chulvi, Ángel Panizo-Lledot, Javier Huertas-Tato, Blanca Calvo-Figueras, M. José Rementería, and Juan Gómez-Romero. Fighting disinformation with artificial intelligence: fundamentals, advances and challenges. *El Profesional de la Información*, 32(3):e320322, ISSN: 1699-2407, 2023. <https://revista.profesionaldelainformacion.com/index.php/EPI/article/view/87328/63436>.
- [9] Wikipedia colaboradores. Gpt-3 — wikipedia, la enciclopedia libre, 2025. <https://es.wikipedia.org/wiki/GPT-3>.
- [10] Wikipedia colaboradores. Gpt-4 — wikipedia, la enciclopedia libre, 2025. <https://es.wikipedia.org/wiki/GPT-4>.
- [11] IBM. ¿qué es un qubit? — ibm, 2025. <https://www.ibm.com/es-es/topics/qubit>.
- [12] Wikipedia colaboradores. Gato de schrödinger — wikipedia, la enciclopedia libre, 2025. https://es.wikipedia.org/wiki/Gato_de_Schr%C3%B6dinger.
- [13] Wikipedia contributors. Entrelazamiento cuántico — wikipedia, la enciclopedia libre, 2024. https://es.wikipedia.org/wiki/Entrelazamiento_cu%C3%A1ntico.
- [14] Wikipedia contributors. Circuito cuántico — wikipedia, la enciclopedia libre, 2024. https://es.wikipedia.org/wiki/Circuito_cu%C3%A1ntico.
- [15] Germán Fernández. Computación cuántica: fundamentos teóricos, estado del hardware y aplicaciones futuras, 2023. <https://germanfernandez.com/computacion-cuantica-fundamentos-teorico-s-estado-del-hardware-y-aplicaciones-futuras-2/>.
- [16] Amit Katwala. Las computadoras cuánticas tienen un grave problema de ruido, 2024. <https://es.wired.com/articulos/computadoras-cuanticas-tienen-problema-de-ruido-computacion-informatica-cuantica>.
- [17] Wikipedia colaboradores. Cuántica de escala intermedia ruidosa — wikipedia, la enciclopedia libre, 2024. https://es.wikipedia.org/wiki/Cu%C3%A1ntica_de_escal_a_intermedia_ruidosa.
- [18] Amelie Schreiber, Riza Velioğlu, Sonja Schimbeno, Jielun Chen, Matthias Klusch, and Patrick Glauner. Quantum natural language processing: A comprehensive survey. *arXiv preprint arXiv:2403.19758*, 2024. <https://arxiv.org/html/2403.19758v2>.
- [19] Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*, 36(1-4):345–384, 2010. <https://arxiv.org/abs/1003.4394>.

- [20] Cambridge Quantum Computing Limited (CQCL). lambeq: A quantum natural language processing toolkit, 2024. <https://github.com/CQCL/lambeq>.
- [21] G. Li, X. Zhao, and X. Wang. Quantum self-attention neural networks for text classification. *arXiv preprint arXiv:2205.05625*, 2022. <https://doi.org/10.48550/arxiv.2205.05625>.
- [22] J. Wei, Z. He, C. Chen, M. Deng, and H. Situ. Psvm-based quantum self-attention neural network. In *2023 International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR)*, pages 1–6. IEEE, 2023. <https://doi.org/10.1109/icwapr58546.2023.10337270>.
- [23] Q. Li, S. Upreti, B. Wang, and D. Song. Quantum-inspired complex word embedding. In *Proceedings of The Third Workshop on Representation Learning for NLP*, 2018. <https://doi.org/10.48550/arxiv.1805.11351>.
- [24] Eduardo Reck Miranda, Pedro Thomas, and Paulo Vitor Itaboraí. Q1synth: A quantum computer musical instrument. *Applied Sciences*, 13(4), 2023. <https://www.mdpi.com/2076-3417/13/4/2386>.
- [25] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, 5(1):103, 2019. <https://www.nature.com/articles/s41534-019-0223-2>.
- [26] Jinkai Tian, Xiaoyu Sun, Yuxuan Du, Shanshan Zhao, Qing Liu, Kaining Zhang, Wei Yi, Wanrong Huang, Chaoyue Wang, Xingyao Wu, Min-Hsiu Hsieh, Tongliang Liu, Wenjing Yang, and Dacheng Tao. Recent advances for quantum neural networks in generative learning. *arXiv preprint arXiv:2206.03066*, 2022. <https://arxiv.org/abs/2206.03066>.
- [27] Wikipedia contributors. Divergencia de kullback-leibler — wikipedia, la enciclopedia libre, 2024. https://es.wikipedia.org/wiki/Divergencia_de_Kullback-Leibler.
- [28] J.E. Johnson. Notes on maximum mean discrepancy, 2024. https://jejjohnson.github.io/research_notebook/content/notes/kernels/mmd.html.
- [29] Marcello Benedetti, Delfina Garcia-Pintos, Oscar Perdomo, Vicente Leyton-Ortega, Yunseong Nam, and Alejandro Perdomo-Ortiz. A generative modeling approach for benchmarking and training shallow quantum circuits. *npj Quantum Information*, 5(1):1–9, 2019. <https://doi.org/10.1038/s41534-019-0157-8>.
- [30] Mohammad H Amin, Evgeny Andriyash, Jason Rolfe, Bohdan Kulchytskyy, and Roger Melko. Quantum boltzmann machine. *Physical Review X*, 8(2):021050, 2018. <https://doi.org/10.1103/PhysRevX.8.021050>.
- [31] Habib Gültekin. Bbc news archive dataset — kaggle, 2025. <https://www.kaggle.com/datasets/hgultekin/bbcnewsarchive>.
- [32] FasterCapital. Función de coste del error cuadrático medio (mse), 2025. [https://fastercapital.com/es/tema/funci%C3%B3n-de-coste-del-error-cuadr%C3%A1tico-medio-\(mse\).html#:~:text=Definici%C3%B3n%3A%20La%20funci%C3%B3n%20de%20costo,reales%20de%20la%20variabla%20objetivo](https://fastercapital.com/es/tema/funci%C3%B3n-de-coste-del-error-cuadr%C3%A1tico-medio-(mse).html#:~:text=Definici%C3%B3n%3A%20La%20funci%C3%B3n%20de%20costo,reales%20de%20la%20variabla%20objetivo).
- [33] José Manuel Ortiz Aguilar, Noé García Espinosa, Omar Ramírez González, and Jorge del Campo Acevedo. Diseño de un circuito generador cuántico entrenable para distribución de probabilidad deseada. In *Congreso Mexicano de Ingeniería Electrónica, Eléctrica y Computación (COMETI)*, 2021. <https://www.riego.mx/congresos/comeii2021/files/ponencias/extenso/COMETI-21005.pdf>.
- [34] Eugenio M. Fernández Aguilar. Silenciando el ruido cuántico: el metamaterial que protege a los cúbits, 2025. <https://www.muyinteresante.com/ciencia/silenciando-ruido-cuantico-metamaterial-cubits.html>.
- [35] Wikipedia contributors. Swap test — Wikipedia, The Free Encyclopedia, 2024. https://en.wikipedia.org/wiki/Swap_test.

- [36] John Wright. Lecture 8: More swap. <https://people.eecs.berkeley.edu/~jswright/quantumlearningtheory24/scribe%20notes/lecture08.pdf>, 2024. UC Berkeley, CS294 - Quantum Learning Theory, Fall 2024.
- [37] Wikipedia contributors. Simulated annealing — wikipedia, the free encyclopedia, 2025. https://en.wikipedia.org/wiki/Simulated_annealing.
- [38] Wikipedia contributors. Holevo’s theorem — wikipedia, the free encyclopedia, 2025. https://en.wikipedia.org/wiki/Holevo%27s_theorem.
- [39] Facebook AI Research. Fasttext, 2024. <https://fasttext.cc/>.
- [40] Dynex [DNX]. How to implement a quantum self-attention transformer on dynex. *Medium*, August 2024. <https://dynexcoin.medium.com/how-to-implement-a-quantum-self-attention-transformer-on-dynex-4c3c72e03eea>.
- [41] Wikipedia colaboradores. Transformada cuántica de fourier — wikipedia, la enciclopedia libre, 2025. https://es.wikipedia.org/wiki/Transformada_cu%C3%A1ntica_de_Fourier.
- [42] Wikipedia colaboradores. Algoritmo de grover — wikipedia, la enciclopedia libre, 2025. https://es.wikipedia.org/wiki/Algoritmo_de_Grover.
- [43] Wikipedia colaboradores. Word2vec — wikipedia, la enciclopedia libre, 2025. <https://es.wikipedia.org/wiki/Word2vec>.
- [44] IBM Cloud. Quantum computing api reference — ibm cloud, 2025. <https://cloud.ibm.com/apidocs/quantum-computing>.
- [45] IBM Quantum. Qiskit — ibm quantum, 2025. <https://www.ibm.com/quantum/qiskit>.
- [46] Samuel Yen-Chi Chen, Shinjae Yoo, and Yao-Lung L. Fang. Quantum long short-term memory. *arXiv preprint arXiv:2009.01783*, 2020. <https://arxiv.org/abs/2009.01783>.
- [47] PennyLane AI. Data reuploading classifier — pennylane documentation, 2024. https://pennylane.ai/qml/demos/tutorial_data_reuploading_classifier.
- [48] PatentPC. Quantum computing energy consumption: How sustainable is it? — latest data, 2025. <https://patentpc.com/blog/quantum-computing-energy-consumption-how-sustainable-is-it-latest-data>.
- [49] The Quantum Insider. Quantum apollo: Interlune plans to mine the moon to power cryogenic technology, 2025. <https://thequantuminsider.com/2025/01/26/quantum-apollo-interlune-plans-to-mine-the-moon-to-power-cryogenic-technology>.
- [50] Iker Marcelo. Pfm-iker-marcelo: Repositorio del proyecto de fin de máster, 2025. <https://github.com/Iker-Marcelo/PFM-Iker-Marcelo/tree/main>.
- [51] Speech Technology Magazine. Voice is poised to take a quantum leap, 2023. <https://www.speechtechmag.com/Articles/Editorial/Features/Voice-Is-Poised-to-Take-a-Quantum-Leap-166696.aspx>.
- [52] FasterCapital. Computación cuántica para el procesamiento del lenguaje natural, 2023. Consultado el 5 de junio de 2025.
- [53] IonQ. Supercharging ai with quantum computing: Quantum-enhanced large language models, 2024. <https://ionq.com/blog/supercharging-ai-with-quantum-computing-quantum-enhanced-large-language>.
- [54] Chang Zhao, Jiawei Song, Xinyi Liang, Jinzhao Liu, Daoyi Ding, and Yang Liu. Quantum natural language processing: A comprehensive survey, 2024. https://www.researchgate.net/publication/381812136_Quantum_Natural_Language_Processing_A_Comprehensive_Survey.
- [55] IonQ. Supercharging ai with quantum computing: Quantum-enhanced large language models, 2024. <https://ionq.com/blog/supercharging-ai-with-quantum-computing-quantum-enhanced-large-language>.

Acrónimos

- **API**: *Application Programming Interface*, punto de entrada que permite al código comunicarse con servicios externos como IBM Quantum.
- **BasisEmbedding**: puerta de PennyLane/Lambeck que codifica un vector clásico en la base computacional de un registro cuántico.
- **BBC**: abreviatura del *British Broadcasting Corporation*, fuente del dataset “BBC News” usado para los experimentos del clasificador/discriminador.
- **BERT**: *Bidirectional Encoder Representations from Transformers*, modelo Transformer clásico citado como inspiración de los mecanismos de auto-atención.
- **CNOT**: puerta *Controlled-NOT* de dos qubits, básica para generar entrelazamiento en varios circuitos del proyecto.
- **CPU**: *Central Processing Unit*; todos los entrenamientos en simulador se corrieron sobre CPU.
- **CRZ**: rotación controlada sobre el eje Z , usada en los ansatz del generador y en el bloque de auto-atención.
- **CZ**: puerta *Controlled-Z*; introduce fase condicional en los generadores de QGAN y QCBM.
- **GAN**: *Generative Adversarial Network*, arquitectura generativa clásica que inspiró las QGAN.
- **GPT**: familia *Generative Pre-trained Transformer* (GPT-3, GPT-4) usada como referencia de modelos de lenguaje a gran escala.
- **GPU**: *Graphics Processing Unit*; alternativa acelerada a la CPU mencionada como futura mejora de cómputo.
- **IBM**: *International Business Machines*, proveedor del hardware cuántico `ibm_sherbrooke` e `ibm_kyiv` utilizado.
- **KL**: divergencia de *Kullback-Leibler*, métrica de disimilitud empleada como función de coste en QCBM.
- **LSTM**: *Long Short-Term Memory*, red recurrente clásica empleada como línea base y dentro del generador híbrido.
- **MMD**: *Maximum Mean Discrepancy*, otra métrica de distancia entre distribuciones usada en los experimentos con QCBM.
- **NISQ**: *Noisy Intermediate-Scale Quantum*, generación actual de hardware cuántico con la que se trabaja.
- **NLP**: *Natural Language Processing*, disciplina marco del proyecto.
- **POVM**: *Positive Operator-Valued Measure*, medición generalizada que mejora la precisión de QSANN.
- **QAE**: *Quantum Amplitude Estimation*, subrutina usada junto con QGAN para fijar precios de opciones.
- **QBM**: *Quantum Boltzmann Machine*, modelo generativo basado en Hamiltonianos cuánticos.
- **QCBM**: *Quantum Circuit Born Machine*, generador de distribuciones empleando la regla de Born.
- **QGAN**: *Quantum GAN* (se usa también en plural **QGANs**), versión híbrida cuántico-clásica de las GAN.
- **QFT**: *Quantum Fourier Transform*, bloque central del módulo de auto-atención cuántica.

- **QLSTM**: versión híbrida cuántica de la LSTM con una capa variacional de 3 qubits.
- **QNLP**: *Quantum Natural Language Processing*, línea de investigación que enmarca todo el TFM.
- **QRNN**: *Quantum Recurrent Neural Network*, alternativa cuántica a las RNN clásicas citada en el estado del arte.
- **QSANN**: *Quantum Self-Attention Neural Network*, versión cuántica del mecanismo de auto-atención.
- **Q1Synth**: sintetizador musical basado en el estado de un solo qubit, referenciado como inspiración de generación cuántica.
- **Qiskit**: SDK de IBM para programar y ejecutar circuitos cuánticos.
- **RAM**: *Random Access Memory*; su límite impidió simular circuitos de 32 qubits en Google Colab.
- **RNN**: *Recurrent Neural Network*, familia a la que pertenecen LSTM y QRNN.
- **RX/RY/RZ**: rotaciones de un solo qubit en los ejes X , Y y Z , parámetros básicos de todos los ansatzes.
- **SA**: *Simulated Annealing*, algoritmo clásico de optimización usado para buscar secuencias válidas en el experimento con SWAP test.
- **Softmax**: función que convierte logits en distribuciones de probabilidad, empleada para decodificar la salida cuántica.
- **SWAP test**: circuito que compara la similitud entre dos estados cuánticos; base del generador por búsqueda clásica.
- **TF-IDF**: *Term Frequency-Inverse Document Frequency*, representación vectorial usada en el clasificador clásico.
- **Word2Vec**: modelo de incrustaciones de palabras usado para inicializar los vectores de 8–32 dimensiones.