KI - Künstliche Intelligenz
https://doi.org/10.1007/s13218-024-00861-w

TECHNICAL CONTRIBUTION

Quantum Natural Language Processing

Dominic Widdows[1] · Willie Aboumrad[1] · Dohun Kim[2] · Sayonee Ray[1] · Jonathan Mei[1]

Received: 19 February 2024 / Accepted: 2 July 2024

Abstract
Language processing is at the heart of current developments in artificial intelligence, and quantum computers are becoming available at the same time. This has led to great interest in quantum natural language processing, and several early proposals and experiments. This paper surveys the state of this area, showing how NLP-related techniques have been used in quantum language processing. We examine the art of word embeddings and sequential models, proposing some avenues for future investigation and discussing the tradeoffs present in these directions. We also highlight some recent methods to compute attention in transformer models, and perform grammatical parsing. We also introduce a new quantum design for the basic task of text encoding (representing a string of characters in memory), which has not been addressed in detail before. Quantum theory has contributed toward quantifying uncertainty and explaining "What is intelligence?" In this context, we argue that "hallucinations" in modern artificial intelligence systems are a misunderstanding of the way facts are conceptualized: language can express many plausible hypotheses, of which only a few become actual.

Keywords Quantum language processing · QNLP · Quantum AI · Quantum string encoding

1 Introduction

In early 2024, quantum computing and AI are two of the most rapidly-moving and talked-about areas of science and technology. The availability of dialog systems based on large language models (LLMs) has raised the profile of natural language processing (NLP) to a historic high, developing and expanding very quickly. This expansion has led to AI models being deployed as systems and introduced as components in new ways, leading to improvements and efficiencies, but also mistakes and concerns. Thus the demand for improvements in AI is at an all-time high, with a renewed focus on reliability and trust.

Quantum theory offers new forms of mathematical modeling, computation and communication. Mathematical models for language operations motivated explicitly by quantum theory have been used in information retrieval [1, 2], logic and disambiguation [3], and language composition [4, 5]. Similar models have been developed in many social sciences and demonstrated successful results over classical alternatives, long before any such models were implemented and run on quantum computers. More abstractly, entire classes of quantum machine learning (ML) models have been theoretically shown to have more expressive power than comparable classical models [6, 7], though this does not guarantee improvements in results more generally [8]. Running basic NLP algorithms on quantum has become possible only in the last few years, with early-stage results reported by [9, 10].

This paper is intended as an introduction to this landscape, for those interested in language processing and quantum computing, but not necessarily specialists in either. Firstly, Sect. 2 gives a brief introduction to quantum gates and circuits. Section 3 continues with an idealized example of how quantum gates could be used to represent a text string of exponential length in a register of qubits, including some caveats and pitfalls. This gives a glimpse of some of

✉ Dominic Widdows
widdows@ionq.com

Willie Aboumrad
aboumrad@ionq.com

Dohun Kim
dohunkim@postech.ac.kr

Sayonee Ray
sayonee@ionq.com

Jonathan Mei
jonathan.mei@ionq.com

1 IonQ, Inc., 4505 Campus Dr, College Park, MD, USA

2 Pohang University of Science and Technology, Pohang, South Korea

the wonder, and some of the challenges, of quantum computing. The main body of the paper surveys ways in which other aspects of language processing have already been modeled on quantum computers, including embedding vectors, sequences, attention, and grammatical structure (Sects. 4–7). This gives a snapshot of of where quantum NLP has got to at this stage of the NISQ era. Finally, Sect. 8 discusses the challenges of choosing and distinguishing between the hypothetical and the actual. This has taken on fresh urgency in AI systems for fact-checking, to avoid mistaking so-called hallucinations for assertions. We note that language models are *designed* to produce both hypothetical and actual statements, and that quantum mechanics is a better starting point than classical mechanics for modeling this.

2 Quantum Computing Basics

In early 2024, quantum computers are real and in regular use, and quantum runtime is offered as-a-service by many companies, via the internet / cloud. This section introduces some of the basic building blocks of quantum computing, from the perspective of a developer designing quantum programs, particularly to run on today's noisy-intermediate scale quantum (NISQ) hardware. The development process involves specifying a register of qubits, and saying what logic gates and measurements should be performed on these qubits.

The material here overlaps with the introduction of [11]. Some familiarity with quantum mechanics, especially Dirac notation, is assumed, so that $|0\rangle$ and $|1\rangle$ are the basis states for a single qubit whose state is represented in the complex vector space $\mathbb{C}^2$, a 2-qubit state is represented in the tensor product space $\mathbb{C}^2 \otimes \mathbb{C}^2 \cong \mathbb{C}^4$ with basis states $|00\rangle, |01\rangle, |10\rangle$ and $|11\rangle$, 3-qubit states are represented in $\mathbb{C}^{\otimes 3} \cong \mathbb{C}^8$ with basis states $|000\rangle, |001\rangle, \dots, |111\rangle$, and so on. For introductions to how linear algebra is written and used in quantum mechanics, see [12, Ch 2]. Quantum measurement is probabilistic: if $|\phi\rangle$ is an eigenvector of a given measurement operator, then a system in the state $|\psi\rangle$ is observed to be in the state $|\phi\rangle$ with probability given by the square of their scalar product, $\langle\phi|\psi\rangle^2$ (the Born rule), and if this outcome is observed, the system is now in the state $|\phi\rangle$.

In mathematical terms, the key features that distinguish quantum from classical computers are superposition and entanglement. Superposition can be realized in a single qubit: the state $\alpha|0\rangle + \beta|1\rangle$ is a superposition of the states $|0\rangle$ and $|1\rangle$, where α and β are complex numbers, with $|\alpha^2| + |\beta^2| = 1$. Each single-qubit logic gate is a linear operator that preserves the orthogonality of the basis states and this normalization condition, and the group of such operators is $U(2)$, the group of complex 2×2 unitary matrices. Single-qubit gates that feature prominently in this paper are

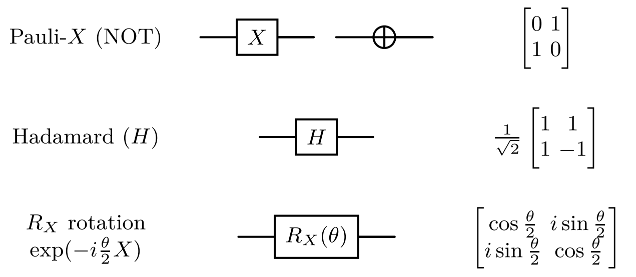

Pauli-X (NOT)
X
$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Hadamard (H)
H
$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
R_X rotation
$\exp(-i\frac{\theta}{2}X)$
$R_X(\theta)$
$\begin{bmatrix} \cos\frac{\theta}{2} & i\sin\frac{\theta}{2} \\ i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}$

Fig. 1 Single-qubit gates used in this paper, and their corresponding matrices, which operate on the superposition state $\alpha|0\rangle + \beta|1\rangle$ written as the column vector $\begin{bmatrix} \alpha & \beta \end{bmatrix}^T$

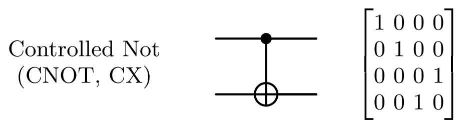

Controlled Not
(CNOT, CX)
$\begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&0&1 \\ 0&0&1&0 \end{bmatrix}$

Fig. 2 Two-qubit CNOT (controlled-X) and gate

shown in Fig. 1. So single-qubit gates coherently manipulate the superposition state of an individual qubit.

Entanglement is a property that connects different qubits. Since the 1930's, quantum entanglement has gone from a hotly-disputed scientific prediction, to a statistical property demonstrated with large ensembles, to a connection created between pairs of individual particles, to a working component in quantum computers. All modern quantum computers have some implementation of an entangling gate, and only one kind is really needed, because all possible 2-qubit entangled states can be constructed mathematically by combining appropriate single-qubit gates before and after the entangling gate. Furthermore, a single 2-qubit entangling gate and a set of single-qubit gates forms a *universal gate-set* for quantum computing [12, §4.5]. Entanglement is the crucial feature that distinguishes quantum computing algorithmically, because predicting the probability distributions that result from quantum operations with entanglement can become exponentially hard for classical computers. In simpler terms, quantum computing is special because it offers special kinds of interference, not because it offers special kinds of in-between-ness.

A quantum circuit consists of a register of qubits, and a sequence of logic gates that act on these qubits. Some of the basic gates used in this paper are shown in Figs. 1 and 2. The Pauli-X gate is commonly used to flip a qubit between the $|0\rangle$ and $|1\rangle$ gates, which is why it is also sometimes called the quantum NOT gate. X-gates applied to different qubits can be used to prepare an input state representing a binary-valued vector: the state $|010 \cdots 001\rangle$ is prepared by applying an X-gate to each of the qubits to be switched to the $|1\rangle$ state.

The Hadamard (H) gate is commonly used to put a qubit into a superposition state: for example, it maps a qubit prepared in the state $|0\rangle$ to the superposition $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Applying an H-gate to each qubit in an array is used to initialize a binary vector all of whose coordinates have a 50-50 chance of being observed in the $|0\rangle$ or $|1\rangle$ state.

Other probabilities, anywhere in the range [0, 1], can be arranged by using fractional rotations (which might involve sending just the same laser-pulse instructions, but for different time periods). An example is given in the $R_X(\theta)$ gate. Several variational quantum algorithms work by gradually optimizing such θ parameters.

The CNOT gate is a 2-qubit entangling gate, that acts upon the state $\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$. In the standard basis, its behavior can be described as "performing a NOT operation on the target qubit if the control qubit is in state $|1\rangle$".

By assembling several 1- and 2-qubit gates, more qubits can become entangled, and we can define *multi-controlled* gates. For example, the 3-qubit Toffoli gate in Fig. 3 flips the lowest qubit if the top qubit is in the $|1\rangle$ state and the middle qubit in the $|0\rangle$ state. It is relatively easy mathematically to start arranging such gate recipes into higher-level operations: for example, the 3-qubit gate acts as a logical AND, from which simple binary arithmetic can be constructed.

However, developers must be careful when using such constructions, because gate complexity and errors can easily build up. In practice, it takes 5 CNOT gates and 9 single-qubit gates to assemble the 3-qubit Toffoli gate, so if gate errors are above 1%, the error-rate in a circuit with just 3 Toffoli gates would be over 50%. In 2024, gate error rates tend to be much better than this, but still, typical circuits today do not reliably run more than a few hundred gates. NISQ-era quantum circuit development tends to tradeoff between sophistication (more gates introduces more tunable parameters) and reliability (fewer gates gives fewer errors). It also leads to designs where classical components are relied upon for many parts of an NLP pipeline, such as storing weights and comparing scores [10].

In quantum machine learning, variational circuits, or parametrized quantum circuits (PQCs), are a particularly clear example of such tradeoffs [13, Ch 5]. Variational circuit designs use i. a quantum circuit with parameters $\{\theta_i\}$, often implemented as variable gate angles, that can be optimized according to a given loss function; and ii. a classical optimizer, responsible for evaluating the measurement outputs of the quantum circuit, and proposing updates to the parameters $\{\theta_i\}$. When both classical and quantum components play such a prominent role, the combination is sometimes called a hybrid system or hybrid workflow.

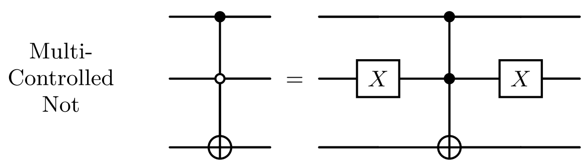


Fig. 3 Three-qubit multi-controlled gate (Toffoli gate) with $|1\rangle$ and $|0\rangle$ control states

The transition from NISQ to fault-tolerant quantum computing will be gradual, and arguably is already underway: recent months have seen encouraging progress in quantum error correction and memory fidelity [14]. This raises long-term expectations that quantum computers will optimize crucial matrix operations, such as solving systems of equations [15], and quantum singular value transformation [16]. However, it is important to remember that even fault-tolerant quantum computing will come with serious caveats, and in particular, quantum components bring no advantage if their I/O costs outweigh their computational gains [17].

Through the rest of this paper, we highlight examples of some of these considerations and design differences as they appear.

3 A Quantum String Encoding Example

Character and string encoding is one of the most basic tasks in language processing, and this section gives a worked example of how this might be performed on a quantum computer using some of the standard quantum circuit and gate patterns introduced in the previous section. This makes a good case-study of some of the promise and challenges of quantum computing, and (as far as we know) is the first such proposal for representing text strings in a quantum computer, comparable to the use of ASCII or Unicode specifications in mainstream classical computing.

To encode a text of meaningful length, this encoding would require many layers of 2-qubit gates, and this design would require fault-tolerant quantum computing, rather than being a NISQ-era proposal. Other methods of encoding the meanings of texts in quantum NLP work have been devised, such as vector embeddings for use in machine learning classifiers, and several such techniques will be surveyed in later sections. The example quantum circuit designs in this section are for the (much older) protocol of representing words as a sequence of characters chosen from a relatively small character set. Some of the quantum word-encoding models based on a sequences and embeddings will be surveyed in later sections of this paper.

The established way to define a string in computer science is to rely on an encoding standard such as ASCII which identifies letters with numbers (A=65, B=66, etc.), and then a string such as *CAB* can be represented as the number sequence [67, 65, 66]. Here the quantum developer faces an immediate and typical challenge: arrays and lists are not standardized components, and a strategy to read from the next location in memory needs to be introduced as part of the design. This lack generalizes: there is much more established literature, software, and hardware for quantum algorithms than for quantum data structures.

One data structure we can use as a building block in quantum circuits is binary positional notation for integers. For example, in a 4-qubit register, the state $|1010\rangle$ could represent the decimal number 10 (if read left-to-right) or 5 (if read right-to-left). This convention was used right at the beginning of quantum computing, in Feynman's proposal of how to build a quantum adder circuit [18], and is part of Shor's integer factoring algorithm [19]. These numbers can be represented in quantum circuits using X-gate bit-flip operations on the corresponding qubits, as in Fig. 4.

Our string encoding design works by entangling a "position" register that records where a character appears, along with an "alphabet" register that says which character appears in that position. So instead of a sequence [3, 1, 2], the string is represented as a tensor product $1_P \otimes 3_C + 2_P \otimes 1_C + 3_P \otimes 2_C$, where n_P is the state representing the P^{th} position, and m_C is the state representing the C^{th} character. A similar pattern is used by [20] for representing images, and the encoding is called QPIXL. QPIXL uses one register to specify the location of a pixel in an image, entangled with another register saying which channel (e.g., red, green, or blue) is being referred to. So, QPIXL also keeps "what it is" and "where it is" in separate registers and entangles these. Emphasizing this similarity, we call our string encoding protocol *QPOSTR*, pronounced "Q-poster", meaning "Quantum Positional String".

The implementation of this formula as a quantum circuit component for encoding the string *cab* is outlined in Fig. 5. The top 2 qubits form the "position" register, and the values on the control qubits are represented by the open and closed circles, $\circ = 0$, $\cdot = 1$. Starting with the top as the least significant bit, the control states are $\circ\circ = 00 = 0$, $\circ\cdot = 01 = 1$, $\cdot\circ = 10 = 2$, etc. The bottom 2 qubits form the "character" or "alphabet" register. Each letter in the alphabet is mapped to a number corresponding to its position, so the gate-recipe for each character is like one of the simple circuits shown in Fig. 4. To encode the string *cab*, a character register of 2 qubits suffices. To encode 26 letters, a character register of 5

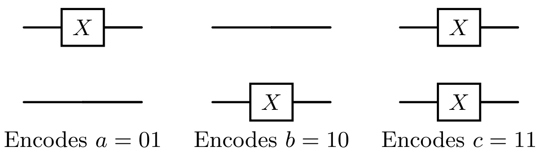


Fig. 4 Simple encodings for a three-letter alphabet in a two-qubit register, using the convention that the top qubit in the register is the least-significant "units" bit in the binary encoding

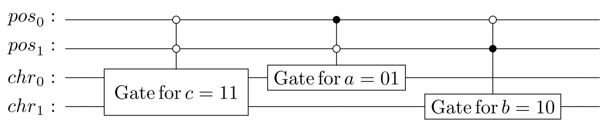


Fig. 5 Position and Character Encoding for the string *cab*

qubits would be required (since $2^5 = 32$), and for the ASCII character set, 7 qubits would be needed.

If the circuit above is prepared in the conventional $|0000\rangle$ state, the first gate controlled on the $\circ\circ = 00$ state is the only one active, and the circuit output will be 00 (in the position register) and 11 (in the character register), saying just "the zeroth character was a c." To prepare a superposition of the characters in all of the positions, the position register is prepared in a uniform distribution over all the available character positions, using a standard array of Hadamard (H) gates. This gives the full circuit for encoding the string *cab* in Fig. 6. Character positions beyond the length of the string are untouched, or left with character "0" in that position. Using the convention that character "0" represents a space, this is equivalent to padding a string with trailing zeros to make its length a power of 2.

With n qubits, the position register can encode up to 2^n positions, and with m qubits, the alphabet register can encode up to 2^m characters. Thus, a QPOSTR circuit with $m + n$ qubits can represent a string of length up to 2^n, with up to 2^m characters. By contrast, a classical computer requires $m \times 2^n$ classical bits to store the same string.

As a thought experiment, we can use GPT-3 training metadata to demonstrate this savings. The size of the training dataset is reported at ~300B tokens [21], so a generous estimate of 12 characters-per-token allows for $12 \times 300 \times 10^9 < 2^{42}$ character-positions, for which the positional encoding fits in 42 qubits. There are currently 149,813 Unicode characters, so even this alphabet fits in 18 qubits, which means that the entire ~45TB training dataset of GPT-3 could fit into a mere $18 + 42 = 60$ qubits!

We can recover information about which character is in which position by adding an output register with the same number of qubits as the character register. The circuit for this is shown in Fig. 7. The multi-controlled gates that connect the QPOSTR representation to the readout register are configured to detect the same position in the string. Each extra qubit in the controls for these gates is set to detect a

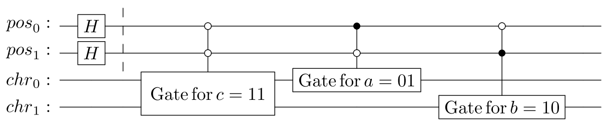


Fig. 6 QPOSTR Encoding for the string *cab*

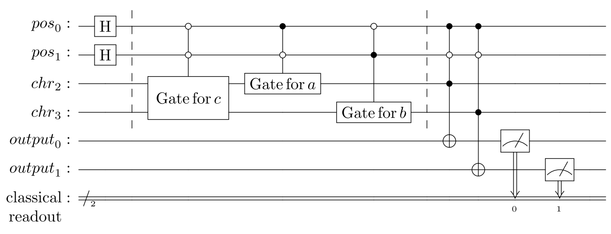


Fig. 7 QPOSTR readout circuit which recovers the character *"a"* from position ∘· = 01 of the string *cab*

particular bit in the character register, and if this bit is set to a 1, then the corresponding output bit is set to a 1.

If this process could be repeated many times, and if the output qubits could be reset to $|0\rangle$ independently of the other qubits in the register, eventually the entire input string or any parts of it could be read out to classical memory. Superficially, this extension of the readout circuit gives a circuit that demonstrates that the whole original string can be recovered from the QPOSTR quantum encoding. In other words, it looks as if we can recover an exponentially-long string (2^n characters) from an encoding that uses $n + \lceil \log_2(\text{alphabet size}) \rceil$ qubits!

However, this is deceptive: due to the H gates at the beginning, we have no way of knowing *which* position will be measured by each readout operation, and to guarantee statistically that each position is sampled, we would need an exponential number of measurements on different copies of the state. (In practice, many shots of the quantum circuit would need to be run.) This is in line with a general theorem in quantum computing, the Holevo bound [12, 12.1.1], which limits the amount of classical information that can reliably be recovered from a quantum state.

Thus, QPOSTR gives an exponential space advantage over the classical alternative. However, it still takes a length of time at least linear in the length of the string to run the quantum circuit that loads the string into quantum memory. Like many quantum encodings, QPOSTR only gives an exponential space saving, which offers no obvious practical advantage without a corresponding time saving. Thus, even if we were to encode the GPT-3 training data, it is unclear at this moment what savings could come from doing so. That said, we optimistically speculate that future algorithms or data structures may be able to better take advantage of this encoding.

More generally, the challenge of preparing a quantum memory that can be maintained and successively queried is sometimes described as research in QROM and QRAM [22, 23]. Minimizing the number of gate operations is a key goal in such work, and the position encoding used in QPOSTR can be regarded as one of the "simple (but suboptimal)" encoding methods described in Fig 3 of [23]. The task they are interested in is encoding electronic spectra, but they also consider encoding "words" as an example toy problem. The extra step that QPOSTR takes is explicitly to map register values to alphabetic characters, which enables such a unitary positional encoding to represent a text as a sequence of alphabetic characters.

Rather than demonstrating a new quantum advantage, the QPOSTR example is intended to showcase some of the excitement, but also some of the gotchas of quantum computing. It is astonishing that an exponentially long string can be encoded like this at all, but once the engineering caveats around that statement are properly understood, we see that the explicit information we can recover from this representation is much smaller.

4 Word Embeddings and Text Classification

Representing words as vectors of coordinates is a technique that goes back at least to the 1960 s and early information retrieval systems [24]. The key theoretical motivation behind such distributional semantics methods is that words that appear in similar contexts tend to have similar meanings [25, 26]. Based on their distribution in text, embedding techniques map words to vector spaces, where their similarity is typically measured by the inner product of their corresponding vectors.

Semantic properties of vectors in lower-dimensional projections were analyzed in the 1990 s [27], and by the early 2000 s, overlaps between the logic of word vectors in information retrieval and state vectors in quantum mechanics had been explicitly recognized [1, 28]. In the past decade, embeddings for classical NLP have jumped from having a resurgence in academia to becoming massively mainstream in industry [29–31]. Naturally, this suggests embeddings could be just as central for QNLP, especially since the mathematics of vectors and tensors has become a common language for both AI and quantum computing [32].

There are many ways to add a quantum flavor to embeddings. In information retrieval, [2] used density matrices and quantum probability to include term-term dependencies in retrieval weighting. Their quantum probability model for bigrams prefigures the more general probabilistic models developed by [33].

Word2ket [34] was introduced as a quantum-inspired solution for compressing embeddings. A tensor network is a decomposition of a high-dimensional tensor into an approximate product of lower-dimensional tensors or vectors. For example, if $M \approx U \otimes V$ then M can be represented using approximately $\dim(U) + \dim(V)$ coordinates, rather than $\dim(U) \times \dim(V)$. (More precisely, in matrix coordinates, $M' \approx uv^T$ for column vectors u and v corresponding to appropriately reshaped U and V, and M' is a reshaped version of M; see [35, 36] for details.) Word2ket uses tensor networks to create low-dimensional approximations for individual word vectors, and entire vocabularies. This mathematical initiative continues: for example, [37] report using tensor networks to compress the parameters of an LLM (LlaMA-2 7B model) to 30% of its original size while retaining over 90% of the original accuracy.

These methods are quantum-inspired, in the sense of drawing deliberately on quantum mathematical models, but running on classical computers.

4.1 Building Quantum Embeddings

Next we discuss techniques relating to embeddings intended for use on actual quantum devices, treating separately the topics of building quantum embeddings and using them. The circuits proposed in this section are intended for NISQ-era rather than fault-tolerant devices.

One of the most well-known recent techniques for building word embeddings is word2vec [29]. Taking inspiration from this line of work, we propose a quantum computing implementation of word2vec.

Word2vec is a group of word embedding methods that use shallow neural networks to capture the semantic properties of words. Word2vec includes two popular methods: Continuous Bag-of-Words (CBOW) and Skip-gram. CBOW and Skip-gram have different ways of learning the word embeddings. CBOW takes the context words around the target word as input and tries to predict the target word. Skip-gram does the opposite: it takes the target word as input and tries to predict the context words. The output of both methods is a probability distribution over all of the words in the vocabulary, which is computed using the softmax function. This can be computationally expensive and impractical when the size of vocabulary is large.

Skip-gram with Negative Sampling (SGNS) [38] is a variant of Skip-gram that reduces computational complexity by simplifying the objective function. Instead of predicting the probability over complete vocabulary, SGNS only tries to distinguish the true context words from a few randomly sampled negative words, which are assumed to be irrelevant to the target word. Thus, the classification problem is simplified from multi-class to binary. The negative sampling procedure also effectively balances the training dataset.

In our implementation, we use quantum states as word vectors, and use quantum fidelity to apply cosine similarity. That is, encoding two words as $|x\rangle$ and $|y\rangle$, we measure their similarity as $|\langle x|y\rangle|^2$ via the swap test [39], as shown in Fig. 8.

One of the challenges of quantum word embedding is how to efficiently load words as quantum states. We consider two potential schemes for embedding words to quantum state: memory-efficient embedding and circuit-efficient embedding.

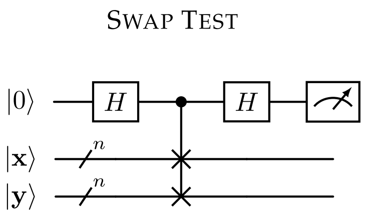


Fig. 8 Swap test circuit, where the probability of measuring a $|0\rangle$ in the top qubit is $|\langle x|y\rangle|$ reflecting the overlap between the $|x\rangle$ and $|y\rangle$ states [39]

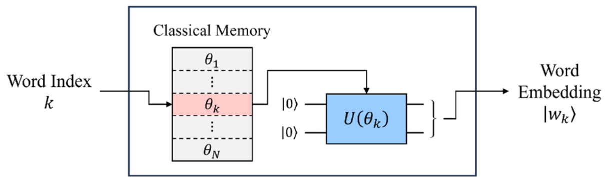


Fig. 9 Circuit-efficient embedding

In memory-efficient embedding, the quantum state of every word in vocabulary of size N is represented by a single unitary operation $U(\theta) \in SU(2^n)$, where $n = \lceil \log_2 N \rceil$ and θ is a set of learnable parameters. The m-qubit quantum state for the k-th word $|w_k\rangle \in \mathbb{C}^{\otimes m}$ is obtained from $U(\theta)$ by applying it to the computational basis state $|k\rangle$ and discarding ancillary $n - m$ qubits.

This scheme allows us to store a large number of words in small number of qubits, which is exponentially efficient in memory usage. However, the resulting quantum circuit that has sufficient expressiveness to implement $U(\theta)$ has exponential depth, making it impractical for circuit-based quantum computation. Moreover, the state preparation process involves post-selection, and is thus non-deterministic due to the probabilistic nature of measurement.

In contrast, the second scheme, circuit-efficient embedding, represents the k-th word by a quantum state of the form $|w_k\rangle := U(\theta_k)|0\rangle \in \mathbb{C}^{\otimes m}$, where $U(\theta_k) \in SU(2^m)$ is a unitary operation parameterized by θ_k, which is specific to each word. This allows us to prepare the quantum state using a depth-efficient circuit in a deterministic process, without using excessive ancillary qubits. While it requires more classical memory to store the parameters, it is more flexible since one can add or remove words from the vocabulary during training.

The circuit-efficient and memory-efficient patterns are depicted in Figs. 9 and 10. In structure, the circuit-efficient pattern is like the word-embedding in word2ket [34], and the memory-efficient pattern is like the word2ketXS whole vocabulary encoding. For word2ket, the motivation for expressing a whole vocabulary in a more entangled tensor network is to reduce classical memory, whereas in

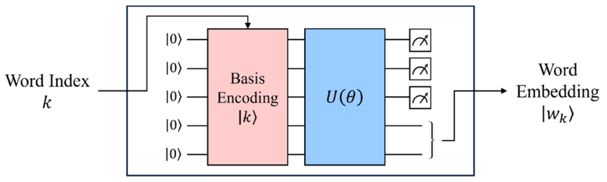


Fig. 10 Memory-efficient embedding

this design, it makes more use of the efficient quantum memory.

To learn the parameters for these embedding schemes, we adapt the classical word2vec methods, CBOW, Skip-gram and SGNS, to the quantum setting. For quantum CBOW and Skip-gram, we introduce a parameterized unitary operator $V(\phi) \in SU(2^n)$ that defines the probability distribution $p_\phi(k|w) := |\langle k|V(\phi)(|w\rangle \otimes |0\rangle)|^2$, where w is either a pooled (e.g. averaged) embedding of context words for CBOW, or the embedding for the target word for Skip-gram. This removes the need of computing the costly softmax function, by using the natural output of the quantum circuit to predict the index among 2^n computational basis states. We note that quantum CBOW, cannot be directly applied for quantum word embeddings, since the direct averaging of quantum states is not a natural operation for quantum computers. Instead, it may be possible to use superpositions of quantum states.

Quantum CBOW and Skip-gram inherit the difficulties of the multi-class problem from the classical version. Both also suffer from the training problem that does not affect the classical versions: barren plateaus [40]. The barren plateau describes the phenomenon where the loss function and its gradient exponentially concentrate as the number of qubits increases. The unitary $V(\phi)$ leads to a barren plateau in the loss landscape, making the training process difficult to scale.

Hence, we propose quantum SGNS, which leverages a simplified structure to mitigate the scaling issue. Quantum SGNS uses the embeddings for two words directly, instead of needing to additionally train $V(\phi)$. Given the target word $|w\rangle$, quantum SGNS tries to maximize the likelihood $p(v|w) := |\langle v|w\rangle|^2$ if v is a context word and minimize it if v is negative sample. By combining quantum SGNS with the circuit-efficient embedding scheme, we enable their practical use on current quantum devices. Future work includes exploring the effects of different similarity kernels on quantum word2vec, extending these circuits to implement word2ket, and understanding algebra in the embedding space.

4.2 Using Quantum Embeddings

Once words are encoded as vectors, these vectors can be used in many machine learning systems, including support vector machines, which can be used for supervised classification. This sometimes involves calculating a *kernel* function, which computes similarities between input vectors, sometimes involving computations that would be intractable if all the coordinates were constructed and compared explicitly [41, Ch 5]. This is regarded as a promising research direction for quantum machine learning, because quantum kernel circuits that compare 2^n coordinates or amplitudes can be implemented using just n qubits [13]. However, as with the QPOSTR string encoding example in Sect. 3, if the number of *gates* still scales with the number of coordinates, the use of a logarithmic number of *qubits* is a saving of space but with no corresponding time advantage.

Several quantum vector encodings or “feature maps” for word embedding vectors were compared by [42], and used for sentiment analysis experiments. The ZZ-feature map was found to be the most successful, achieving a classification accuracy of 62% on classification experiments involving small test sets of roughly 10K words each. This result showed initial promise, and was the largest quantum text experiment reported to-date, but also indicates how small today’s quantum NLP experiments, are compared with even modest-sized classical NLP systems.

One additional use case for embeddings is in factual grounding and retrieval, which we discuss in more detail in Sect. 8.

5 Sequential Models for Text Generation

Quantum generative modeling is still a largely unexplored area of opportunity, with many unsolved challenges. In some cases, the data is too large and is partitioned into segments that are correlated but treated as independent for sake of computation [43]. In others, the term is used to describe settings in which a quantum circuit is used as a discrete source of randomness within an otherwise classical neural network that memorizes a select few data samples [44].

In NLP specifically, [33] describes how to model the joint distribution $p(X_0, X_1)$ of a given a set of bigrams (x_0, x_1) and compute the marginal distributions $p(X_0)$ and $p(X_1)$ using linear algebra operations native to quantum computing (here we use capital letters to denote random variables and lowercase to denote particular values that the random variable can take). [10] follows this theory to implement a Quantum Circuit Born Machine (QCBM) [45] to learn the joint distribution $p(X_0, X_1)$. While the QCBM can efficiently sample pairs from $p(X_0, X_1)$ or marginals $p(X_0)$ and $p(X_1)$, generating text sequentially from this model requires sampling from the conditional $p(X_1|X_0 = x_0)$. This requires discarding samples for which $X_0 \neq x_0$, which can become prohibitive when scaling to larger vocabularies, and especially for rare prefix words x_0. In addition, the bigram model forgoes a hidden state like those found in Recurrent Neural Networks (RNNs) that can learn to represent longer dependencies. Thus, while this model can easily sample bigrams directly, it is not optimized for the task of sampling longer sequences.

A class of Bayesian network models, called n-gram models, have been successful in multiple language processing tasks, including information retrieval, text generation, and part-of-speech tagging [46]. However, they suffer from poor performance and generalizability issues in sparse data regimes and fail to capture nonlocal syntactic relations. Handling out-of-vocabulary words or resolving ambiguity also pose challenges as n-grams do not have built-in semantic understanding [47]. A Hidden Markov models (HMM) is a Markov model whose output from any given state is probabilistic rather than deterministic, which hides the internal state. A canonical example is when the hidden states are part-of-speech tags, such N(oun) or V(erb), which generate explicit words with given probabilities [48, §9.2].

The feed forward and recurrent neural networks are specific instances of the HMM. However, HMMs also face challenges in estimating accurate probabilities when the context size increases or when the correlations get too long in languages. Beyond language processing, HMMs have been widely used in other scientific fields as well. In [49], the authors discuss probabilistic models, particularly HMMs and their derivatives, and their applications in biological sequence analysis. Language models, particularly those developed for aligning and comparing sequences, can be designed to recognize patterns in biological sequences, infer evolutionary relationships, and identify functional elements.

Recently, quantum techniques are being explored along this direction due to their potential in capturing long-range correlations. [50] introduced a quantum enhanced version of the HMM, named the basis-enhanced Bayesian Circuit, which leverages quantum contextuality and non-locality to boost the expressivity of classical HMMs. They developed a minimal quantum extension of the bigram HMM, by incorporating measurements in a Bayesian circuit in multiple bases. They demonstrated improved performance of the quantum enhanced model in certain sequential datasets, including one containing DNA sequences with nonlocal structures. This leads to many interesting questions about the potential and utility of similar quantum methods in natural language processing tasks. Particularly, if quantum properties like contextuality and non-locality still give a provable advantage in terms of model expressivity when processing and extracting semantic meaning from a long sequence of words or protein structures.

Building toward longer sequences, [51] train a quantum classifier for predicting the topic of sentences and describe a classical scheme for using the classifier to perform conditional generation (that can also be used on classical classifiers). Since it does not actually train a standalone quantum generative model, this method suffers from a similar problem of also needing to discard many samples from a base model in order to generate one sample from an induced model, though the correlated editing-based annealing scheme could be guided to be more efficient than independently sampled shots from a QCBM. [52], shown in Fig. 11, proposes a framework for a quantum RNN that can be used to build an actual generative model to autoregressively produce text. This autoregressive modeling allows for dealing with correlations across segments of larger data, addressing the problem from [43]. However, the architecture, while expressive, is far too expensive for current hardware on non-trivial problem sizes. [53] perform sequence classification on actual hardware as shown in Fig. 12. Unfortunately, the architecture they propose is not powerful enough to perform autoregressive modeling.

We explore how to bridge some of the gap between general but expensive sequence processing of [52] and the currently-achievable but underpowered architecture of [53]. To achieve this, we use the paradigm of [54] to combine the power of nonlinear Multi-Layer Perceptrons (MLPs) with the inherent randomness of quantum computing to directly

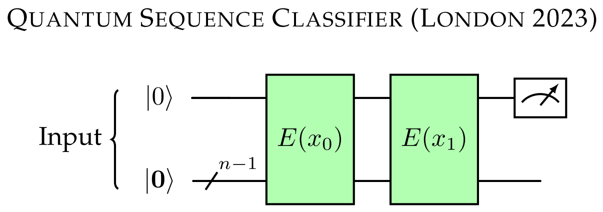


Fig. 12 London et al. (2023) circuit. The input is the sequence (x_0, x_1). Only the first qubit is measured. The output is the probability that the sequence is in class 1

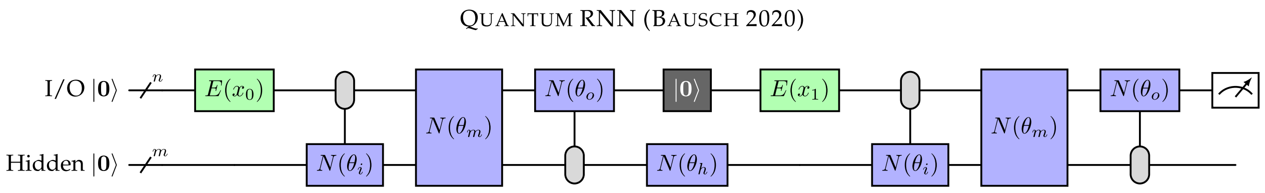


Fig. 11 Bausch (2023) circuit. The input is the sequence (x_0, x_1). The mixing and hidden blocks are prohibitive for current hardware. The output is the next token in the sequence as the outcome from a single shot

sample from rich classes of probability distributions. Compared to [52], our proposed architecture, shown in Fig. 13, drops the use of the reset operation and uses identity mixing, similar to [53].

In the figures, the details of the quantum neurons have been abstracted away to highlight the main similarities and differences between the methods. The text labeling on the left denotes the purpose of the register. $E(x)$ denotes a block of gates for encoding inputs x, $N(\theta)$ denotes a nonlinear neuron parameterized by variables θ as in [55], and the dark gray gate with the $|0\rangle$-ket is a reset operation. The parameter subscript i denotes input, o denotes output, h denotes hidden, and m denotes mixing. The light gray vertical rounded rectangle denotes that the qubits in the register are being used as control for the corresponding neuron as a sequence of single-control gates, not as true multi-control gates. We depict the architectures from [52] and [53] with a sequence length of 2 with input sequence (x_0, x_1), and the Multi-Layer Perceptron (MLP) from [54] with a single hidden layer.

In simulation, the model is trained using backpropagation from gradients computed from noiseless state vector simulation. The model produces the probability distribution over the 11 words in the vocabulary corresponding to the word that the model predicts comes next after the observed sequence. In actual implementation, the model would be trained using backpropagation from estimated gradients e.g. via the parameter-shift rule [56]. The model would produce for each shot a sample corresponding to the index of a single predicted word.

Our proposed architecture is evaluated in a small-scale noiseless simulation. We consider a dataset of 7 sentences using a vocabulary of 11 unique words. We compare our proposal against two baseline models: one random uniform prediction model and one inspired by [53]. We compare performance between models trained on 5 sentences by evaluating on the remaining 2 sentences their perplexity [57], for which a lower score indicates better performance. A naive uniform random prediction on this dataset yields a perplexity of 11. Using a 9-qubit [53] model with 297 parameters, we achieve a perplexity of 8.15. Using a 9-qubit model that we propose with 172 parameters, we achieve a perplexity of 2.79.

To our knowledge, this is the first fully quantum sequential text generation architecture that is designed with the capabilities and limitations of current NISQ-era devices in mind. Our simulation results demonstrate the viability of the approach for implementation on actual hardware while achieving a reasonable level of perplexity.

6 Attention in Quantum NLP Models

So far we have discussed models for studying sentences as word / token sequences. Making such models scale to longer sequences has always been a challenge: with n-gram models, the value of n has always been small [48, Ch 6]; and RNN architectures including LSTMs, while accurate for short sequences, had trouble scaling to cover long-range dependencies [41, Ch 15].

6.1 Attention in Classical LLMs

Attention is designed to address this problem. The attention methodology was used to enhance an RNN sequence model for machine translation by [58], which enabled the model to capture longer-range relationships as well. Although it still relied on the encoder-decoder paradigm, the bidirectional RNN architecture introduced in [58] features a distinct context vector for each word in the sentence. Each context vector depends on a sequence of annotations which contains information about the entire sentence with a strong focus on the parts of the sentence surrounding the context vector's associated input word. The annotations are weighted according to an alignment model, which scores how well an output token matches inputs around a given position.

This approach further was developed further by [59], demonstrating a system where transformer blocks incorporating attention, layer norm, multi-layer perceptrons, and residual connections, fully replace recursive units. This Transformer model — centered around *scaled dot-product attention* — made previous RNN-based encoder-decoder architectures obsolete when it demonstrated improved performance on various translation tasks.

Importantly, [59] adapted the Transformer architecture for use in text generation. Their model is auto-regressive, and at each step it consumes the previously generated symbols as additional input when producing new text. In addition, it is worthy to note that the Transformer was later adapted to the setting of computer vision, where it outperformed

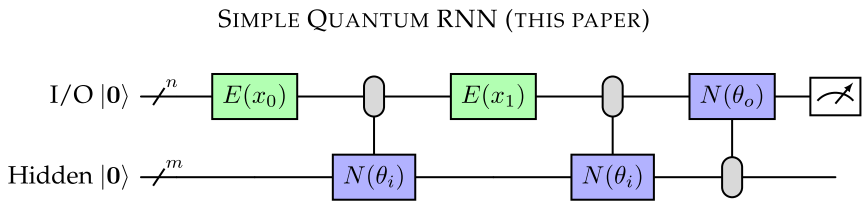


Fig. 13 Proposed circuit. The input is the sequence (x_0, x_1). The output can be the next token in the sequence as the outcome from a single shot

state-of-the-art convolutional neural networks in various image classification challenges [60].

In general, "an attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and outputs are all vectors" representing embedded tokens; the output is a weighted sum of the values, with the weights measuring the compatibility between corresponding query and key [59]. *Self-attention* refers to computing attention coefficients intra-sequence, i.e., on the same input sequence. A key feature of self-attention layers is that they provide a mechanism for different tokens in the input sequence to interact, thereby allowing models to infer contextual information about individual tokens by weighing the importance of pairwise interactions; in other words, how much attention a given input token should pay to every other token in the sequence.

In particular, the "Scaled Dot-Product" attention layer featured in [59] computes the dot-products of the query with all the keys, normalizes according to the dimension of the query and key vectors, and then applies the softmax function to obtain the weights of all the pairs, which are the values. Rather than enumerate all the indices for summation, it is typical to write the lists of vectors as matrices, whereby the definition takes the common form

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V,$$

where d is the embedding dimension, Q, K, and V are matrices of size wd where w is the number of words / tokens in a sequence, and the softmax function $x_i \to e^{x_i} / \sum_i(e^{x_i})$ is applied to each row. This formulates dot product attention as a matrix multiplication (time $O(w^2d)$), a softmax step (time $O(w^2)$), and a final matrix multiplication (time $O(wd^2)$).

A key advantage enjoyed by the Transformer over the previous RNN architectures is that this multiplication can be parallelized, which computes the pairwise relationships between all the tokens in a sequence at once. In addition, the computational cost does not depend on the distance between tokens in the sequence, as in previous models. Together, these properties accounted for a drastic reduction in training time over sequential RNN models. The main drawback is that the computational complexity still scales quadratically in the number of tokens in a given sequence (roughly the number of words in a sentence). The problem of approximating or providing an alternative to self-attention with sub-quadratic complexity spawned its own burgeoning research field [61, 62].

6.2 Near-term Quantum Self-attention Mechanisms

In hopes of improving this quadratic scaling, and since attention layers have become so successful as key components in state-of-the-art models for NLP tasks, various quantum approaches have been suggested. This section focuses on near-term quantum circuit designs.

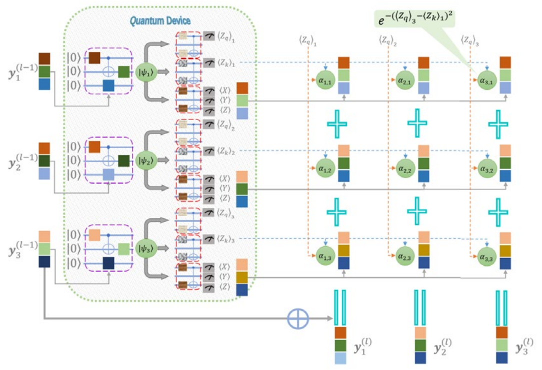


Fig. 14 The Quantum Self-Attention Neural Network (QSANN) architecture proposed in [63]. The network features various consecutive self-attention layers. At the $(l-1)$st layer, the classical feature vectors $y_k^{(l-1)}$ are encoded into a high-dimensional qubit state space (circuits boxed in purple). The process is repeated three times. Then parametrized ansatze, with gate layout as in Fig. 15, representing the query, key, and value transformations are applied (circuits boxed in red). The resulting states are measured and various expectation values are computed to produce the classical query, key, and value vectors. These are sent to a classical device for processing, where weights are computed using a Gaussian kernel and the results are averaged to obtain final attention coefficients

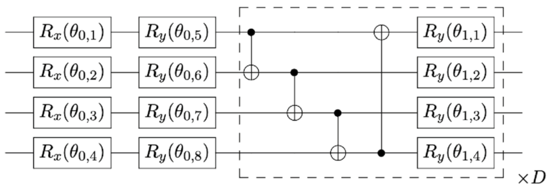


Fig. 15 Parametrized ansatz implementing the query, key, and value transformations in [63]'s QSANN

A quantum self-attention network called QSANN is implemented by [63], who claim it is the first of these. By mapping encoded feature vectors into a high-dimensional Hilbert space using a quantum circuit, QSANN aims to extract correlations that are intractable classically. For an illustration see Fig. 14. First they construct an encoder circuit to load classical feature vectors onto an n-qubit quantum state; they use one classical feature vector for each token in the input sequence. The number of qubits n is a hyperparameter that should be adjusted as relevant to available hardware. Next they apply parametrized quantum circuits, with identical gate layouts but different parameter values in order to compute the query, key, and value vectors for each classical feature vector. The circuit layout is illustrated in

Fig. 15. At this stage the query, key, and value vectors are encoded as quantum states, so measurements must be made in order to extract useful information; the resulting query and key are the expectation values of the Pauli-Z operators applied to the first qubit of the resulting states, and the value is a vector of expectation values of various Pauli operators. Attention scores are then computed on a classical device as a weighted average of the output values. Interestingly, [63] introduce a Gaussian kernel to compute the weights on the values vector; they claim the Gaussian kernel can more easily correlate quantum states with little overlap, which is needed, e.g., if two tokens are closely related in a sentence but their quantum state embeddings happen to be distant in the qubit state space. The proposal of [63] still requires quadratic classical computation, and its main source of quantum advantage relies on using efficiently processing vectors in high-dimensional Hilbert space to unearth hidden relationships between embedded tokens.

The work of [64] builds on these ideas, seeking quantum advantage in the same vein. By introducing various sets of ancilla qubits, the authors obviate the need to perform intermediate measurements during the attention computation. (This could be thought of as a more sophisticated example of the ancilla readout qubits pattern used in the QPOSTR design of Sect. 2.) In this modality, query, key, and value quantum state-vectors are computed by applying parametrized ansatze and swapping onto ancilla registers sequentially, as shown in Fig. 16. Compatibility between query and keys is computed by a *Quantum Logical Similarity (QLS)* module, which is implemented as a sequence of Toffoli and CNOT gates, as shown in Fig. 17. This is a key step: it computes the overlap between query on keys directly on the quantum device, thereby improving on [63].

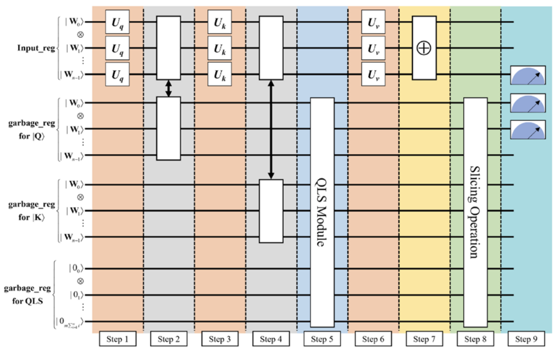


Fig. 16 The Quantum Self-Attention Network (QSAN) introduced in [64]. This architecture uses ancilla qubits to hold intermediate results and proposes computing the attention coefficients entirely on the quantum processor, obviating the need for intermediate measurements. In addition, it features a slicing operation to reduce the number of measurements required

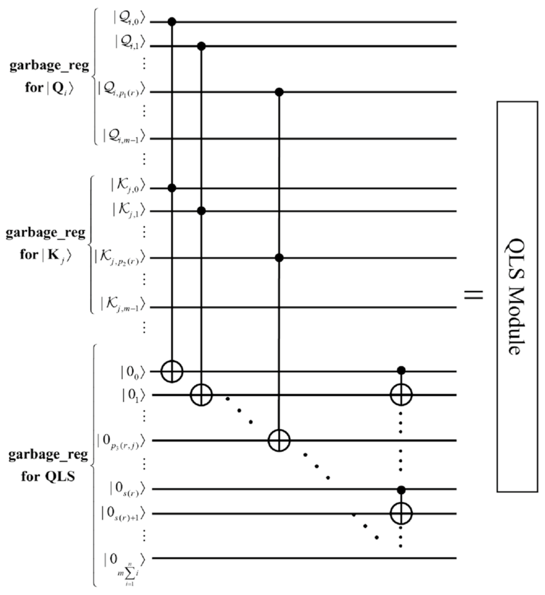
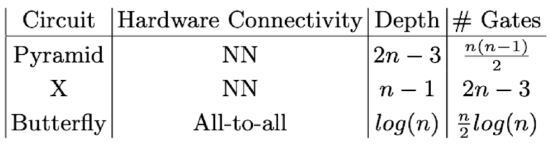


Fig. 17 The Quantum Logic Similarity (QLS) module proposed in [64], implemented as a sequence of Toffoli and CNOT gates

While these two proposals address quantum self-attention mechanisms in QNLP directly, [65] proposes one for use in Vision Transformers for image classification, seeking quantum advantage in reducing the computational cost of the scaled dot-product attention calculation. Concretely, the authors introduce so-called *orthogonal layers* to compute compatibility scores between query and keys on the quantum hardware; these layers efficiently implement parametrized transformations on encoded feature vectors, as described in Fig. 18. The main novelty here is that [65] use the unary encoding circuit to encode token feature vectors into the Hamming weight-1 subspace of the qubit state space. This encoding is advantageous because their orthogonal layers preserve the subspace, and they can be used to compute dot-products between query and keys in logarithmic time, assuming quantum gates can be applied in parallel. [65] report preliminary results from simulation, and with a 6-qubit quantum processor.



Circuit	Hardware Connectivity	Depth	# Gates
Pyramid	NN	$2n-3$	$\frac{n(n-1)}{2}$
X	NN	$n-1$	$2n-3$
Butterfly	All-to-all	$log(n)$	$\frac{n}{2}log(n)$

Fig. 18 Computational complexity of the dot-product compatibility between query and keys using circuits with parallel two-qubit gates as proposed by [65]'s quantum Vision Transformer

6.3 Attention on Fault-Tolerant Quantum Computers

The role of attention in systems like GPT [21] has spurred more ambitious proposals, including the recent preprints of [66] and [67], which describe large-scale versions of full transformer-inspired processes on fault-tolerant error-correcting quantum computers. [67] use quantum signal processing and singular value transformation techniques to apply a polynomial approximate softmax, whereas [66], like [64], replace softmax with a linear quantum alternative.

A main challenge for such proposals remains quantum I/O, which these papers acknowledge, but do not solve explicitly. To extract all the weights from a matrix stored in a quantum state would typically require many more shots than there are weights. A smaller alternative is to output token indices directly. A natural NISQ-inspired suggestion is to adapt the token index generation approaches from Sect. 5 to transformer prototypes. While adding a token-generation head may be useful for circumventing I/O issues, the complexity and depth of the remaining bodies of the circuits in these proposals puts practical implementations beyond the NISQ era.

A quantum decoder, for optimized search through a much larger space of long proposed token sequences, is proposed by [68], which casts the problem as probabilistically branching tree-search, which is mathematically equivalent to probabilistic grammar parsing. Even though the hardware capabilities for such quantum operations are still some years away, this connects optimizations in sequence generation from today's softmaxed probabilities, with traditional syntactic approaches to language modeling, which we discuss next.

7 Syntactic Parsing and Logical Forms

The use of general AI techniques such as RNNs and attention has fueled much recent success with NLP, partly because it has enabled much cross-fertilization between language and other kinds of data such as images, audio, and graphs. There are also more traditional NLP techniques, based on grammatical structures found particularly in human language.

Natural languages (and artificial programming languages) express many structured relationships that go beyond proximity in a sequence model. For example, in the sentence "Kim kicked the ball into the goal from halfway down the pitch, right past the goalkeeper, and scored", the grammatical structure of English enables us to infer easily that the person who scored is Kim, in spite of there being 16 words (including 4 other nouns) in between the noun *Kim* and the verb *scored*. Language grammar and syntax studies how these relationships are expressed and structured in different languages, and this field powerfully influenced much of theoretical and computational linguistics during the second-half of the 20th century, particularly through the work of [69, 70]. In such a framework, syntax is the central generative system, on which other aspects of language like phonology and semantics depend [71, Ch 5].

During the 21st century, the reliance of NLP techniques on grammatical rules has declined. This is partly due to the success of statistical and machine-learned models that share methods with other data-intensive areas of AI, and perhaps because the increasing preponderance of informal text created on smartphones immediately contradicts any assumption that the input to an NLP system should consist of distinct grammatical sentences. In computational terms, best-performing parsing algorithms have included the CYK-parser which is worst-case $O(n^3)$, while the more general attention mechanism discussed in the previous section has a baseline $O(n^2)$ performance, which robustly enables *more* resources to be targeted at important long-range relationships. Hence in most large-scale NLP systems today, a grammatical parser is not an explicit component: and with the challenge of reducing computational costs below quadratic, there would need to be a strong reason for requiring the extra burden of a cubic computing step for any large-scale model. When grammatical parsing *is* discussed, it is often as a historical challenge that LLMs can solve quite effectively, not a contemporary challenge for building language models in the first place [72].

However, the use of grammatical parsers has been reintroduced in parts of *quantum* NLP, motivated especially by a mathematical correspondence between the compositional rules of tensors and categorial grammars demonstrated by [5]. This framework is implemented in the `lambeq` system of [73] which is particularly designed for quantum computers [9]. In this system, a grammatical parser prepares a parse-tree from a sentence, which is structurally mapped to a tensor network which is compiled into a quantum circuit: so the quantum circuit encodes the grammatical dependencies of the sentence, assuming its input data consists of uniquely-parsed sentences.

The parsing problem itself is also an interesting challenge for quantum computing, for combinatoric reasons. Computationally, the problem can be phrased as taking a list of words as input, and returning a parse-tree, which is a data structure saying how the elements of the sentence are grouped together, and what grammatical role they play. The two possible tree-structures for a 3-word sentence are shown in Fig. 19. For the simple case of a 3-word sentence, there are only 2 possible trees, one for the structure [[*A B*] *C*] and one for the structure [*A* [*B C*]]. For a 4-word sentence, the five possible trees are are shown in Fig. 20. The number of possible parse trees for a sentence of length

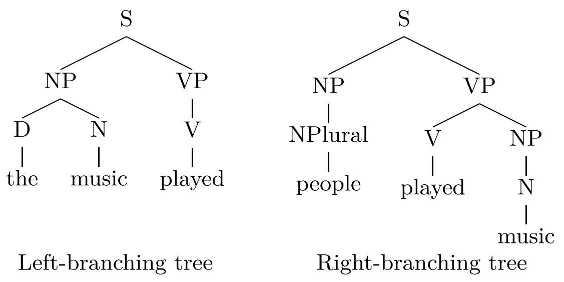


Fig. 19 The simplest nontrivial tree-parsing challenge is to distinguish between the two distinct branching options for a tree with 3 leaf nodes. S = Sentence, N = Nouns, V = Verb, D = Determiner, P = Phrase

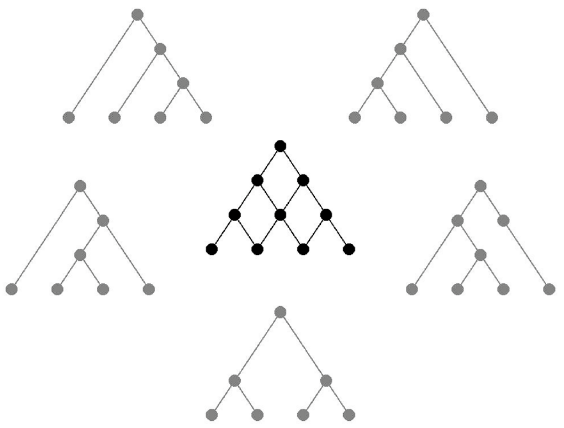


Fig. 20 The five possible binary-branching trees over a 4-item sequence, each of which can be constructed by deleting unwanted branches and nodes from the connected lattice-graph in the center

n grows exponentially and is given by the *Catalan number* $C_n = \frac{1}{n+1}\binom{2n}{n}$.

Ideally, a parser will find an exact parse that accounts for the role of each constituent in the sentence, represented by a single tree-structure. Grammatical rules such as “A sentence in English can be made of a noun phrase subject followed by a verb phrase” can be represented as recipes like S → (NP, VP), and a probabilistic phrase-structure grammar (PCFG) may include many such rules, along with probabilities or weights learned from training data. Various methods for parsing exist, relying on dynamic programming and probabilistic techniques [46, Ch 17]. Multiple parses are often possible, because (for example) prepositional phrases may attach in various locations. The goal is to find the parse tree that maximizes the overall probability of the parse, computed from the combined probability of all the rules used in derivation.

The depiction in Fig. 20 emphasizes the parsing problem as a combinatoric challenge. The search for a best parse looks like a kind of minimal spanning tree problem, which might be formulated as an Ising model, amenable to quantum optimization [74], if the weights or costs for visiting each node could be derived from the PCFG.

However, a key challenge with the parsing problem is that we start with labels only for the leaf nodes, and until there is some hypothesis for the which internal nodes represent which syntactic chunks, there are no estimates for the weights on internal links. (This is part of the motivation for the use of dynamic programming in classical parsers.) Combinatorically, the problem is not exactly to find a minimum spanning tree, because internal nodes that do not correspond to distinct grammatical phrases or *constituents* do not need to be visited. (Moreover, the initial graph is directed: this precludes solutions where the leaf nodes are all visited by a zigzag path along the bottom, because upward steps are forbidden.)

Hence, quantum parsing does not appear to fit one of the known techniques of quantum combinatorial optimization, but the problems are tantalizingly similar, and we hope this framing of the challenge helps the language parsing problem to capture the interest of quantum researchers.

One potential benefit of quantum parsing is that a quantum system may represent more nuanced proposals than just returning a single best parse, or even a classical mixture (estimated probability distribution over different discrete parse proposals). Rather than insisting that all language inputs must pass a parsing test before being processed, a syntactic parser that exchanges quantum information with semantic components in parallel may become a different kind of asset in more parallel architectures.

In another interesting combination of quantum mathematics and natural language syntax, it has been shown that tensor product networks can encode grammatical structure more effectively than LSTMs for generating image captions [75]. Tensor product networks have also been used to construct an attention mechanism from which grammatical structure can be recovered by unbinding role-filler tensor compositions [76].

8 Facts and Language Generation

Throughout its history, quantum theory has motivated new insights on probability and randomness, how the potential and actual are related, and this has led to proposed models for various aspects of human behavior and consciousness [77, 78]. Rapid advancements in customer-facing AI systems, particularly conversational dialog systems supported by large language models (LLMs), have raised many questions about how such systems should be built and used, what should be expected of them, and about whether they exhibit conscious behavior. This section discusses some of the concerns and work in this area, including how quantum theory addresses the difference between hypothetical and actual reality.

One of the key complaints about current LLMs is their propensity to *hallucinate*, or produce sentences with factually false information that still correctly adhere to grammatical rules. While this behavior is in line with what generative models are statistically designed to do, in practice it goes against the public expectation of the AI agent as an all-knowing oracle, especially when manifested as an interactive, question-answering chat interface. Thus, several lines of research have emerged to address the usability issues caused by hallucination. In this section, we first review some of these research directions, and then take a step back to see how quantum computing relates to the philosophical issues that arise in understanding and using LLMs.

Two prominent realms of research are chain-of-thought (CoT) prompting and retrieval augmented generation (RAG). CoT, while often used in the context of complex reasoning tasks [79], is believed to produce more self-consistent results and indeed can be improved by explicitly encouraging self-consistency [80]. However, in the setting of factuality, errors early in the chain may yield incorrect results even with consistent reasoning later in the chain, and the explanation of reasoning provided by the chain may not be correct [81].

Factual grounding via RAG may avoid some issues still present in CoT, often at the cost of either additional training or larger number of inferences. RAG can be largely classed into two paradigms, *a priori* and *post-hoc*, though there is not reason that these techniques could not be combined or even performed iteratively in a loop. *A priori* RAG is explored in [82, 83] using learned embeddings stored in a database and accessed with a learned neural retriever. Scaling up the database and shrinking the language model is explored by [84], matching state-of-the-art performance and illustrating that the world knowledge of a LLM is more separable from its linguistic ability than previously demonstrated. Together, this line of work suggests separability and knowledge base scaling as one possible path forward for utilizing word embeddings to reduce hallucination. While these *a priori* methods show promise, they modify the generation pipeline and require additional training.

Post-hoc text editing methods [85–87] are seeing interest for use with LLMs in part due to the resources required to even fine-tune modern LLMs, let alone pre-train them from scratch. For example, [88] uses the abilities of pre-trained language models and existing information retrieval systems to edit and verify generated text, but requires running several rounds of inference on the base LLM. This trades off a zero fixed cost of using a pre-trained model for a higher variable cost of inference.

In such paradigms, one natural place where quantum computing has potential to enhance performance is in accessing and designing the knowledge base. Grover's algorithm [89] yields a quadratic speedup over classical methods for searching a database, and hence forms the backbone of quantum information retrieval systems [90]. Going one step further, [91] propose using Grover's search to inform the design of the database itself, and specifically target vector databases as the intended application. However, they do not directly compare the quadratic speedup with the complexities achievable by using efficient classical data structures, and a potential interesting direction in this area is the question of how (if possible) to design a practical hybrid knowledge base that combines the best of both classical and quantum processing.

Although techniques such as retrieval-based and knowledge-based generation are a new area in the present-day context of fixing LLMs, methods such as that of [84] hearken back to an older class of designs where the facts are stored in a knowledge base, and the language model is effectively a source of templates, not of facts. The spotlight on language generation in the past few years has refocused work on such methods, and how best to combine them with LLMs [92]. For example, in Fig. 21, a knowledge base is used to find the variable that satisfies the question "When was J.S. Bach born?" (the answer being "1685"), and then a language model is used to express this as the sentence "J.S. Bach was born in the year 1685." (A standard early use of such designs was in mail merging, where a template for a message is combined with a list of different names to generate personalized messages.) Such a language model can just as easily generate the sentence "J.S. Bach was born in 1985", not because it's hallucinating, but because it's working correctly with a different knowledge base.

More generally, probabilistic language models are designed to note that *Wednesday* and *Thursday* are similar, and so having seen the phrase "Let's meet on Wednesday", the model should judge the phrase "Let's meet on Thursday" to be similarly plausible. Saying that such a probabilistic model "hallucinates" when it generates an untrue sentence reflects a fundamental misunderstanding of probabilistic models. Sampling from a probabilistic model is like rolling dice: if we previously observe a 3 and the dice-roll gives a 4, the dice aren't hallucinating a 4 instead of the "true" value of 3. The problem lies in the assumption that plausible

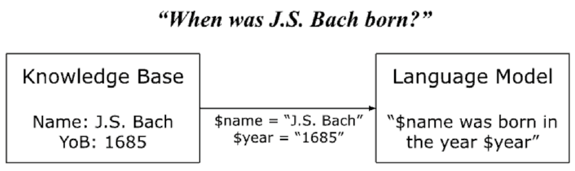


Fig. 21 A traditional division of responsibilities between a knowledge base and a language model that cooperate in generating the answer "J.S. Bach was born in 1685."

probabilistic samples of *language* should correspond to *facts* at all.

The assumption that language expertise and factual reliability *should* go together is easy to make, especially since a significant amount of actual knowledge is conveyed both explicitly and implicitly through writing. In the phrase "Dave beat John", we might ask "Which Dave and which John?" before assessing its truthfulness: but sometimes words take on fixed unique meanings in particular situations, so that if someone says "Caesar beat Pompey", we automatically assume they mean two particular people from the 1st century BC, and if they said "Pompey beat Caesar", that would be considered untrue. However, speaking strictly in the sense of language modelling, a model that also generates "Pompey beat Caesar" sometimes as well as "Caesar beat Pompey", is arguably *better*, because it generates a more comprehensive variety of perfectly fluent and plausible sentences.

The practice of generating text from *just* a language model was popularized by successful machine translation systems [58]. With machine translation, it makes sense that the system is *not* responsible for factual accuracy, because this is the user's responsibility. In concrete terms, a correct translation of "J.S. Bach was born in 1985" from English to German might be "J.S. Bach wurde 1985 geboren", not "Input error: prompt contains factual inaccuracy." Gradually models such as GPT demonstrated that a whole range of prompts, not just translation targets, could elicit plausible and fluent responses [21]. The Chomskian program claimed grammatical fluency as the heart of language decades ago: today, we are seeing that this fluency is another aspect of human behavior that computers can mimic effectively; and the ability to assemble erudite text has become one of the most impressively-solved parts of AI, sometimes leading to problems elsewhere.

Quantum theory intersects with these topics even more fundamentally, by explicitly distinguishing the possible from the actual. A quantum circuit has many possible outcomes that could be observed, but only one outcome is observed when measured: and this fixes the hypothetical situation so that the same outcome is observed next time. A multiplicity of possibilities can become a single fixed event [93]. Formal similarities between this process and language ambiguity were noted by [94], and the quantum economic theory of [95] is based on the use of quantum information to model beliefs about values, and classical information to model amounts of money agreed in fixed transactions.

The problem of distinguishing things that *might* happen from things that *do* happen was behind some of the controversies of early classical mechanics. Leibniz discussed the notion of possible worlds, and maintained that there must be a rational necessity behind (God's) choosing this world [96]. Newton's belief in absolute space implied a fixed zero-point or origin, and Leibniz argued that this implied that God must have made an arbitrary choice without a necessary reason, which was unacceptable [97]. Such considerations of necessity vs. contingency and their relationship to past, present, and future in time, go back at least to the famous sea-fight discussion in Aristotle's *De Interpretatione* [98].

The notion that there are different possible worlds where a macroscopic event did or did not happen, that one of those worlds is chosen based on a small local decision, and this possible world thus *becomes* the actual world, was thrust into the limelight by quantum mechanics. The implication of superposition and large-scale randomness was troubling to Einstein ("God does not play dice!") and Schrödinger, whose famous paradoxical cat was designed to illustrate the absurdity of quantum mechanics in large-scale reality, where "the working of an organism requires exact physical laws" [99]. By contrast, Bohr and Heisenberg supported the Copenhagen Interpretation, where the wave-function represents real possibilities, and *"the transition from the "possible" to the "actual" takes place during the act of observation"* [93, Ch 3].

Some of the challenges inherent in large stochastic problems, like weather forecasting, are thus philosophically related to key questions of how one possible future is selected and becomes the past. Quantum mechanics does not completely answer this question, but it does better than classical mechanics, where the assumption of a deterministic universe avoids the problem. Heisenberg's analysis of where different uncertainties come from, and how we should think about them, has useful insights including *"This probability function represents a mixture of two things, partly a fact and partly our knowledge of a fact"* [93, Ch 3]. This does not tell us how to fix language models, but it is a good reminder that our ways of stating and communicating facts are entirely human. Practically, it helps to understand probabilistic language models as generators of hypothetical utterances, rather than factual statements, and the generative nature of language models is precisely what enables them to go smoothly from data they encountered to data they might just as well have encountered. In a sense, a large language generator is a kind of hypothesis-generator with the gift of the gab. Language models do this task very well, but this should never have convinced us that a model will generate truthful language without an independent source of knowledge. With these considerations, quantum theory has some insight on potential solutions to improve language modeling systems, and at least guards against mistakes that arise from over-deploying hypothesis-generation systems without suitable observation processes.

9 Conclusion

We have taken a whirlwind tour of the state of quantum NLP, seeing the potential and limitations of using quantum computers for understanding language. While we recognize this overview, like any other, cannot be fully comprehensive, we hope that it is nonetheless useful for both the theoretician and practitioner alike.

We reviewed fundamentals of gate-based quantum computing, and from here moved into understanding how these low-level structures and concepts can be used to efficiently encode basic units of language, i.e. text. From there, we built into progressively higher-level concepts, roughly following the hierarchy found in classical NLP.

Through this journey, we have seen how the current scale of applications for quantum NLP on actual hardware has not yet matched that of classical computing techniques. However, quantum methods being developed at the small scale show promise for use on intermediate scale problems as hardware continues progressing, and quantum models that have been shown to be more expressive than their analogous classical counterparts hold potential at large scales.

In the meantime, methods from quantum theory continue to inform AI. During the 2010 s, vectors and tensors became a common mathematical toolset permeating AI, and the adaptation of tensor network methods for scalability continues this theme. We have especially focused on the topical problems that current classical LLMs face. Here, quantum theory has much philosophical guidance to offer on the issues of assessing factuality and sequential inference.

References

1. Van Rijsbergen CJ (2004) The Geometry of Information Retrieval. Cambridge University Press, Cambridge
2. Sordoni A, Nie JY, Bengio Y (2013) Modeling Term Dependencies with Quantum Language Models for IR. In: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '13, p 653–662
3. Widdows D, Peters S (2003) Word vectors and quantum logic. In: Proceedings of the Eighth Mathematics of Language Conference, Bloomington, Indiana
4. Clark S, Pulman S (2007) Combining symbolic and distributional models of meaning. In: AAAI Spring Symposium: Quantum Interaction, p 52–55
5. Coecke B, Sadrzadeh M, Clark S (2010) Mathematical foundations for a compositional distributional model of meaning. CoRR, Preprint at arXiV:1003.4394
6. Coyle B, Mills D, Danos V, Kashefi E (2020) The Born supremacy: quantum advantage and training of an Ising Born machine. npj Quant Inf 6(1):1–11. https://doi.org/10.1038/s41534-020-00288-9
7. Yu Z, Chen Q, Jiao Y, Li Y, Lu X, Wang X, Yang JZ (2023) Provable Advantage of Parameterized Quantum Circuit in Function Approximation. https://doi.org/10.48550/arXiv.2310.07528, http://arxiv.org/abs/2310.07528, arXiv:2310.07528 [quant-ph]
8. Bowles J, Ahmed S, Schuld M (2024) Better than classical? the subtle art of benchmarking quantum machine learning models. Preprint at arXiv:2403.07059
9. Lorenz R, Pearson A, Meichanetzidis K, Kartsaklis D, Coecke B (2023) Qnlp in practice: Running compositional models of meaning on a quantum computer. J Artif Intell Res 76:1305–1342
10. Widdows D, Alexander A, Zhu D, Zimmerman C, Majumder A (2024) Near-term advances in quantum natural language processing. Ann Math Artif Intell, p 1–24
11. Widdows D, Bhattacharyya A (2024) Quantum financial modeling on noisy intermediate-scale quantum hardware: Random walks using approximate quantum counting. Quantum Economics and Finance
12. Nielsen MA (2016) Chuang I (2002) Quantum computation and quantum information. Cambridge University Press Edition, Cambridge
13. Schuld M, Petruccione F (2021) Machine Learning with Quantum Computers. Springer, Cham
14. Bravyi S, Cross AW, Gambetta JM, Maslov D, Rall P, Yoder TJ (2024) High-threshold and low-overhead fault-tolerant quantum memory. Nature 627(8005):778–782
15. Harrow AW, Hassidim A, Lloyd S (2009) Quantum algorithm for linear systems of equations. Phys Rev Lett 103(15):150502
16. Gilyén A, Su Y, Low GH, Wiebe N (2019) Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In: Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, p 193–204
17. Aaronson S (2015) Read the fine print. Nat Phys 11(4):291–293
18. Feynman RP (1985) Quantum mechanical computers. Opt. News 11(2):11–20
19. Shor PW (1994) Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th annual symposium on foundations of computer science, Ieee, p 124–134
20. Amankwah MG, Camps D, Bethel EW, Van Beeumen R, Perciano T (2022) Quantum pixel representations and compression for n-dimensional images. Sci Rep 12(1):7712
21. Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S, Herbert-Voss A, Krueger G, Henighan T, Child R, Ramesh A, Ziegler D, Wu J, Winter C, Hesse C, Chen M, Sigler E, Litwin M, Gray S, Chess B, Clark J, Berner C, McCandlish S, Radford A, Sutskever I, Amodei D (2020) Language models are few-shot learners. Adv Neural Inf Process Syst 33:1877–1901
22. Giovannetti V, Lloyd S, Maccone L (2008) Quantum random access memory. Phys Rev Lett 100(16):160501
23. Babbush R, Gidney C, Berry DW, Wiebe N, McClean J, Paler A, Fowler A, Neven H (2018) Encoding electronic spectra in quantum circuits with linear t complexity. Phys Rev X 8(4):041015
24. Salton G, McGill M (1983) Introduction to modern information retrieval. McGraw-Hill, New York
25. Wittgenstein L (1953) Philospphical Investigations. Blackwell, Blackwell, 3rd edition, 2001
26. Firth J (1957) A synopsis of linguistic theory 1930-1955. Studies in Linguistic Analysis, Philological Society, Oxford
27. Landauer T, Dumais S (1997) A solution to Plato's problem: the latent semantic analysis theory of acquisition. Psychol Rev 104(2):211–240
28. Widdows D (2004) Geometry and meaning. CSLI Publications, Stanford
29. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arxiv. https://doi.org/10.48550/ARXIV.1301.3781
30. Bridgwater A (2023) The Rise Of Vector Databases. https://www.forbes.com/sites/adrianbridgwater/2023/05/19/the-rise-of-vector-databases/, section: Cloud
31. Metinko C (2023) Pinecone Hits $750M Valuation As AI Heats Up Vector Database Market. https://news.crunchbase.com/ai-robotics/startup-venture-funding-database-pinecone/
32. Widdows D, Kitto K, Cohen T (2021) Quantum mathematics in artificial intelligence. J Artif Intell Res 72:1307–1341
33. Bradley TD (2020) At the interface of algebra and statistics. City University of New York, New York

34. Panahi A, Saeedi S, Arodz T (2019) word2ket: Space-efficient word embeddings inspired by quantum entanglement. Preprint at arXiv:1911.04975
35. Hitchcock FL (1927) The Expression of a Tensor or a Polyadic as a Sum of Products. J Math Phys 6(1–4):164–189. https://doi.org/10.1002/sapm192761164
36. Van Loan CF (2000) The ubiquitous Kronecker product. J Comput Appl Math 123(1):85–100. https://doi.org/10.1016/S0377-0427(00)00393-9
37. Tomut A, Jahromi SS, Singh S, Ishtiaq F, Muñoz C, Bajaj PS, Elborady A, del Bimbo G, Alizadeh M, Montero D, et al. (2024) CompactifAI: Extreme compression of large language models using quantum-inspired tensor networks. Preprint at arXiv:2401.14109
38. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013b) Distributed representations of words and phrases and their compositionality. Adv Neural Inf Process Syst 26
39. Barenco A, Berthiaume A, Deutsch D, Ekert A, Jozsa R, Macchiavello C (1997) Stabilisation of quantum computations by symmetrisation. SIAM J Comput 26(5):1541–1557. https://doi.org/10.1137/S0097539796302452
40. McClean JR, Boixo S, Smelyanskiy VN, Babbush R, Neven H (2018) Barren plateaus in quantum neural network training landscapes. Nat Commun 9(1):4812
41. Géron A (2019) Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media,
42. Alexander A, Widdows D (2022) Quantum text encoding for classification tasks. In: 2022 IEEE/ACM 7th Symposium on Edge Computing, p 355–361
43. Huang HL, Du Y, Gong M, Zhao Y, Wu Y, Wang C, Li S, Liang F, Lin J, Xu Y, Yang R, Liu T, Hsieh MH, Deng H, Rong H, Peng CZ, Lu CY, Chen YA, Tao D, Zhu X, Pan JW (2021) Experimental quantum generative adversarial networks for image generation. Phys Rev Appl 16(2):024051. https://doi.org/10.1103/PhysRevApplied.16.024051. arXiv:2010.06201 [quant-ph]
44. Rudolph MS, Toussaint NB, Katabarwa A, Johri S, Peropadre B, Perdomo-Ortiz A (2022) Generation of high-resolution handwritten digits with an ion-trap quantum computer. Phys Rev X 12(3):031010. https://doi.org/10.1103/PhysRevX.12.031010. (**publisher: American Physical Society**)
45. Benedetti M, Garcia-Pintos D, Perdomo O, Leyton-Ortega V, Nam Y, Perdomo-Ortiz A (2019) A generative modeling approach for benchmarking and training shallow quantum circuits. NPJ Quant Inf 5(1):1–9. https://doi.org/10.1038/s41534-019-0157-8
46. Jurafsky D, Martin JH (2023) Speech and Language Processing (3rd Edition draft). Stanford, California, https://web.stanford.edu/~jurafsky/slp3/
47. Bengio Y, Ducharme R, Vincent P (2000) A neural probabilistic language model. In: Leen T, Dietterich T, Tresp V (eds) Advances in Neural Information Processing Systems, vol 13. MIT Press, Cambridge
48. Manning CD, Schütze H (1999) Foundations of Statistical Natural Language Processing. The MIT Press, Cambridge
49. Durbin R, Eddy SR, Krogh A, Mitchison G (1998) Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press, Cambridge
50. Gao X, Anschuetz ER, Wang ST, Cirac JI, Lukin MD (2022) Enhancing generative models via quantum correlations. Phys Rev X 12(2):021037
51. Karamlou A, Pfaffhauser M, Wootton J (2022) Quantum natural language generation on near-term devices. arXiv. https://doi.org/10.48550/arXiv.2211.00727
52. Bausch J (2020) Recurrent Quantum Neural Networks. In: Advances in Neural Information Processing Systems, Curran Associates, Inc., vol 33, pp 1368–1379, https://proceedings.neurips.cc/paper/2020/hash/0ec96be397dd6d3cf2fecb4a2d627c1c-Abstract.html
53. London C, Brown D, Xu W, Vatansever S, Langmead CJ, Kartsaklis D, Clark S, Meichanetzidis K (2023) Peptide binding classification on quantum computers. Quant Mach Intell. https://doi.org/10.48550/arXiv.2311.15696
54. Gili K, Sveistrys M, Ballance C (2023) Introducing nonlinear activations into quantum generative models. Phys Rev A 107(1):012406
55. Cao Y, Guerreschi GG, Aspuru-Guzik A (2017) Quantum Neuron: an elementary building block for machine learning on quantum computers. arXiv. https://doi.org/10.48550/arXiv.1711.11240
56. Schuld M, Bergholm V, Gogolin C, Izaac J, Killoran N (2019) Evaluating analytic gradients on quantum hardware. Phys Rev A 99(3):032331. https://doi.org/10.1103/PhysRevA.99.032331. (**publisher: American Physical Society**)
57. Jelinek F, Mercer RL, Bahl LR, Baker JK (1977) Perplexity-a measure of the difficulty of speech recognition tasks. J Acoust Soc Am 62(S1):S63. https://doi.org/10.1121/1.2016299
58. Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. Preprint at arXiv:1409.0473
59. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Adv Neural Inf Process Syst, p 5998–6008
60. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, et al. (2020) An image is worth 16x16 words: Transformers for image recognition at scale. ICLR / Preprint at arXiv:2010.11929
61. Zaheer M, Guruganesh G, Dubey KA, Ainslie J, Alberti C, Ontanon S, Pham P, Ravula A, Wang Q, Yang L, Ahmed A (2020) Big Bird: Transformers for Longer Sequences. Adv Neural Inf Process Syst 33:17283–17297
62. Poli M, Massaroli S, Nguyen E, Fu DY, Dao T, Baccus S, Bengio Y, Ermon S, Re C (2023) Hyena hierarchy: Towards larger convolutional language models. International Conference on Machine Learning https://openreview.net/forum?id=1sxiBaGEtg
63. Li G, Zhao X, Wang X (2022) Quantum self-attention neural networks for text classification. Preprint at arXiv:2205.05625
64. Zhao Rx, Shi J, Zhang S (2022) QSAN: A near-term achievable quantum self-attention network. Preprint at arXiv:2207.07563
65. Cherrat EA, Kerenidis I, Mathur N, Landman J, Strahm M, Li YY (2022) Quantum vision transformers. Preprint at arXiv:2209.08167
66. Liao Y, Ferrie C (2024) Gpt on a quantum computer. Preprint at arXiv:2403.09418
67. Guo N, Yu Z, Agrawal A, Rebentrost P (2024) Quantum linear algebra is all you need for transformer architectures. Preprint at arXiv:2402.16714
68. Bausch J, Subramanian S, Piddock S (2021) A quantum search decoder for natural language processing. Quant Mach Intell 3(1):16
69. Chomsky N (1957) Syntactic structures. Mouton de Gruyter, The Hague
70. Chomsky N (1965) Aspects of the Theory of Syntax, vol 11. MIT Press, Cambridge
71. Jackendoff R (2002) Foundations of Language. Oxford Universiry Press, Oxford
72. Min B, Ross H, Sulem E, Veyseh APB, Nguyen TH, Sainz O, Agirre E, Heintz I, Roth D (2023) Recent advances in natural language processing via large pre-trained language models: a survey. ACM Comput Surv 56(2):1–40
73. Kartsaklis D, Fan I, Yeung R, Pearson A, Lorenz R, Toumi A, de Felice G, Meichanetzidis K, Clark S, Coecke B (2021) lambeq: An Efficient High-Level Python Library for Quantum NLP. Preprint at arXiv:2110.04236

74. Lucas A (2014) Ising formulations of many np problems. Front Phys 2:5
75. Huang Q, Smolensky P, He X, Deng L, Wu D (2017) Tensor product generation networks for deep NLP modeling. Preprint at arXiv:1709.09118
76. Huang Q, Deng L, Wu D, Liu C, He X (2019) Attentive tensor product learning. Proc AAAI Conf Artif Intell 33:1344–1351
77. Busemeyer JR, Bruza PD (2012) Quantum models of cognition and decision. Cambridge University Press, Cambridge
78. Atmanspacher H (2020) Quantum Approaches to Consciousness. In: Zalta EN (ed) The Stanford Encyclopedia of Philosophy, Summer, 2020th edn. Stanford University, Stanford
79. Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, Chi E, Le QV, Zhou D (2022) Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. Advances in Neural Information Processing Systems 35:24824–24837, https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html
80. Wang X, Wei J, Schuurmans D, Le Q, Chi E, Narang S, Chowdhery A, Zhou D (2023) Self-Consistency Improves Chain of Thought Reasoning in Language Models. https://doi.org/10.48550/arXiv.2203.11171, http://arxiv.org/abs/2203.11171, arXiv:2203.11171 [cs]
81. Turpin M, Michael J, Perez E, Bowman SR (2023) Language Models Don't Always Say What They Think: Unfaithful Explanations in Chain-of-Thought Prompting. https://doi.org/10.48550/arXiv.2305.04388, http://arxiv.org/abs/2305.04388, arXiv:2305.04388 [cs]
82. Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, Kuttler H, Lewis M, Wt Yih, Rocktaschel T, Riedel S, Kiela D (2020) Retrieval-augmented generation for knowledge-intensive NLP tasks. Adv Neural Inf Process Syst 33:9459–9474
83. Guu K, Lee K, Tung Z, Pasupat P, Chang M (2020) Retrieval Augmented Language Model Pre-Training. In: Proceedings of the 37th International Conference on Machine Learning, PMLR, pp 3929–3938, https://proceedings.mlr.press/v119/guu20a.html, iSSN: 2640-3498
84. Borgeaud S, Mensch A, Hoffmann J, Cai T, Rutherford E, Millican K, Driessche GBVD, Lespiau JB, Damoc B, Clark A, Casas DDL, Guy A, Menick J, Ring R, Hennigan T, Huang S, Maggiore L, Jones C, Cassirer A, Brock A, Paganini M, Irving G, Vinyals O, Osindero S, Simonyan K, Rae J, Elsen E, Sifre L (2022) Improving Language Models by Retrieving from Trillions of Tokens. In: Proceedings of the 39th International Conference on Machine Learning, PMLR, pp 2206–2240, https://proceedings.mlr.press/v162/borgeaud22a.html, iSSN: 2640-3498
85. Thorne J, Vlachos A (2021) Evidence-based Factual Error Correction. In: Zong C, Xia F, Li W, Navigli R (eds) Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Association for Computational Linguistics, Online, pp 3298–3309, https://doi.org/10.18653/v1/2021.acl-long.256, https://aclanthology.org/2021.acl-long.256
86. Balachandran V, Hajishirzi H, Cohen WW, Tsvetkov Y (2022) Correcting diverse factual errors in abstractive summarization via post-editing and language model infilling. arXiv. https://doi.org/10.48550/arXiv.2210.12378
87. Schick T, Dwivedi-Yu J, Jiang Z, Petroni F, Lewis P, Izacard G, You Q, Nalmpantis C, Grave E, Riedel S (2022) PEER: A Collaborative Language Model. https://doi.org/10.48550/arXiv.2208.11663, http://arxiv.org/abs/2208.11663, arXiv:2208.11663 [cs]
88. Gao L, Dai Z, Pasupat P, Chen A, Chaganty AT, Fan Y, Zhao V, Lao N, Lee H, Juan DC, Guu K (2023) RARR: Researching and Revising What Language Models Say, Using Language Models. In: Rogers A, Boyd-Graber J, Okazaki N (eds) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Toronto, Canada, pp 16477–16508, https://doi.org/10.18653/v1/2023.acl-long.910, https://aclanthology.org/2023.acl-long.910
89. Grover LK (1996) A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96, ACM Press, Philadelphia, Pennsylvania, United States, pp 212–219, https://doi.org/10.1145/237814.237866, http://portal.acm.org/citation.cfm?doid=237814.237866
90. Giri PR, Korepin VE (2017) A review on quantum search algorithms. Quant Inf Process 16(12):315. https://doi.org/10.1007/s11128-017-1768-7
91. Pronin CB, Ostroukh AV (2023) Synthesis of quantum vector databases based on grovers algorithm. arXiv. https://doi.org/10.48550/arXiv.2306.15295
92. Zhang H, Song H, Li S, Zhou M, Song D (2023) A survey of controllable text generation using transformer-based pre-trained language models. ACM Comput Surv 56(3):1–37
93. Heisenberg W (1958) Physics and philosophy: The revolution in modern science. Vladimir Djambov
94. Widdows D (2003) A mathematical model for context and word-meaning. In: Fourth International and Interdisciplinary Conference on Modeling and Using Context, Stanford, California
95. Orrell D (2020) Quantum Economics and Finance: An Applied Mathematics Introduction. Panda Ohana Publishing, New York
96. Rescher N (1996) Leibniz on possible worlds. Studia Leibnitiana pp 129–162
97. Bouquiaux L (2008) Leibniz against the unreasonable Newtonian physics. In: Bouquiaux L (ed) Leibniz: What Kind of Rationalist? Springer, Cham, pp 99–110
98. McKeon R (ed) (1941) The Basic Works of Aristotle. Random House
99. Schrödinger E (1944) What is life? Cambridge University Press (with mind and matter and autobiographical sketches, 1996)