Quantum Machine Intelligence (2023) 5:10
https://doi.org/10.1007/s42484-023-00097-1

RESEARCH ARTICLE

Grammar-aware sentence classification on quantum computers

Konstantinos Meichanetzidis[1,2] · Alexis Toumi[1,2] · Giovanni de Felice[1,2] · Bob Coecke[1,2]

Received: 26 May 2022 / Accepted: 6 January 2023 / Published online: 14 February 2023

Abstract
Natural language processing (NLP) is at the forefront of great advances in contemporary AI, and it is arguably one of the most challenging areas of the field. At the same time, in the area of quantum computing (QC), with the steady growth of quantum hardware and notable improvements towards implementations of quantum algorithms, we are approaching an era when quantum computers perform tasks that cannot be done on classical computers with a reasonable amount of resources. This provides an new range of opportunities for AI, and for NLP specifically. In this work, we work with the Categorical Distributional Compositional (DisCoCat) model of natural language meaning, whose underlying mathematical underpinnings make it amenable to quantum instantiations. Earlier work on fault-tolerant quantum algorithms has already demonstrated potential quantum advantage for NLP, notably employing DisCoCat. In this work, we focus on the capabilities of noisy intermediate-scale quantum (NISQ) hardware and perform the first implementation of an NLP task on a NISQ processor, using the DisCoCat framework. Sentences are instantiated as parameterised quantum circuits; word-meanings are embedded in quantum states using parameterised quantum-circuits and the sentence's grammatical structure faithfully manifests as a pattern of entangling operations which compose the word-circuits into a sentence-circuit. The circuits' parameters are trained using a classical optimiser in a supervised NLP task of binary classification. Our novel QNLP model shows concrete promise for scalability as the quality of the quantum hardware improves in the near future and solidifies a novel branch of experimental research at the intersection of QC and AI.

Keywords String diagrams · Compositionality · Natural language processing · Quantum computing

1 Introduction

NLP is a rapidly evolving area of AI of both theoretical importance and practical interest (Jurafsky and Martin 2000; Blackburn and Bos 2005). Large language models, such as the relatively recent GPT-3 with its 175 billion parameters (Brown et al. 2020), show impressive results on general NLP tasks and one dares to claim that humanity is entering Turing-test territory (Turing 1950). Such models, almost always based on neural networks, work by learning to model conditional probability distributions of words in the context of other words. The textual data on which they are trained are mined from large text corpora; the probability distribution to be learned captures the statistical patterns of cooccurrence of words in the data. Due to this, in this work we shall refer to such models as *distributional*.

NLP technology becomes increasingly entangled with everyday life as part of search engines, personal assistants, information extraction and data-mining algorithms, medical diagnoses, and even bioinformatics (Searls 2002; Zeng et al. 2015). Despite success in both language understanding and language generation, under the hood of mainstream NLP models one exclusively finds deep neural networks, which famously suffer the criticism of being uninterpretable black boxes (Buhrmester et al. 2019).

One way to bring transparency to said black boxes is to explicitly incorporate *linguistic structure*, such as *grammar* and *syntax* (Lambek 1958; Montague 2008; Chomsky 1957), into distributional language models. Note, in this work we will use the terms grammar and syntax interchangeably, the essence of these terms being that they refer to structural information that the textual data could be supplemented with. A prominent approach attempting this merge is the *Distributional Compositional Categorical* model of natural language meaning (DisCoCat) (Coecke et al. 2010; Grefenstette and

✉ Konstantinos Meichanetzidis
k.mei@quantinuum.com

1 Quantinuum, 17 Beaumont Street, Oxford, OX1 2NA, UK

2 Department of Computer Science, University of Oxford, Oxford, OX1 3QD, UK

Sadrzadeh 2011; Kartsaklis and Sadrzadeh 2013), which pioneered the paradigm of combining explicit grammatical (or syntactic) *structure* with distributional (or statistical) methods for encoding and computing meaning (or semantics). There has also been follow-up related work on neural-based models where syntax is also incorporated in a recursive neural network, where the syntactic structures dictate the order of the recursive calls to the recurring cell (Socher et al. 2013). This approach also provides the tools for modelling linguistic phenomena such as lexical entailment and ambiguity, as well as the transparent construction of syntactic structures like, relative and possessive pronouns (Sadrzadeh et al. 2013; 2014), conjunction, disjunction, and negation (Lewis 2020).

From a modern lens, DisCoCat, as it is presented in the literature, is a *tensor network language model*. Recently, the motivation for designing interpretable AI systems has caused a surge in the use of tensor networks in language modelling (Pestun and Vlassopoulos 2017; Gallego and Orus 2019; Bradley et al. 2019; Efthymiou et al. 2019). A tensor network is a graph whose vertices are endowed with tensors. Every vertex has an arity, i.e. a number of edges to which it belongs which represent the tensor's indices. Edges represent tensor contractions, i.e. identification of the indices joined by the edge and summation over their range (Eistein summation). Intuitively, a tensor network is a compressed representation of a multilinear map. Tensor networks have been used to capture probability distributions of complex many-body systems, both classical and quantum, and they also have a formal appeal as rigorous algebraic tools (Eisert 2013; Orús 2019).

Quantum computing (QC) is a field which, in parallel with NLP is growing at an extremely fast pace. The prominence of QC is now well-established, especially after the experiments aiming to demonstrate quantum advantage for the specific task of sampling from random quantum circuits (Arute et al. 2019). QC has the potential to reach the whole range of human interests, from foundations of physics and computer science, to applications in engineering, finance, chemistry, and optimisation problems (Bharti et al. 2021), and even procedural map generation (Wootton 2020).

In the last half-decade, the natural conceptual fusion of QC with AI, and especially the subfield of AI known as machine learning (ML), has lead to a plethora of novel and exciting advancements. The quantum machine learning (QML) literature has reached an immense size considering its young age, with the cross-fertilisation of ideas and methods between fields of research as well as academia and industry being a dominant driving force. The landscape includes using quantum computers for subroutines in ML algorithms for executing linear algebra operations (Harrow et al. 2009), or quantising classical machine learning algorithms based on neural networks (Beer et al. 2020), support vector machines, clustering (Kerenidis et al. 2019), or artificial agents who learn from interacting with their environment (Dunjko et al. 2016), and even quantum-inspired and dequantised classical algorithms which nevertheless retain a complexity theoretic advantage (Chia et al. 2020). Small-scale classification experiments have also been implemented with quantum technology (Havlíček et al. 2019; Li et al. 2015).

From this collection of ingredients, there organically emerges the interdisciplinary field of quantum natural language processing (QNLP), a research area still in its infancy (Zeng and Coecke 2016; O'Riordan et al. 2020; Wiebe et al. 2019; Bausch et al. 2020; Chen 2002), combines NLP and QC and seeks novel quantum language model designs and quantum algorithms for NLP tasks. Building on the recently established methodology of QML, one imports QC algorithms to obtain theoretical speedups for specific NLP tasks or use the quantum Hilbert space as a feature space in which NLP tasks are to be executed.

The first paper on QNLP, using the DisCoCat framework by Zeng and Coecke (Zeng and Coecke 2016), introduced an approach where a standard NLP task are instantiated as quantum computations. The task of sentence similarity was reduced to the closest-vector problem, for which there exists a quantum algorithm providing a quadratic speedup, albeit assuming a Quantum Random Access Memory (QRAM). The mapping of the NLP taks to a quantum computation is attributed to the mathematical similarity of the structures underlying DisCoCat and quantum theory. This similarity becomes apparent when both are expressed in the graphical language of string diagrams of monoidal categories or process theories (Coecke and Kissinger 2017). The categorical formulation of quantum theory is known as Categorical Quantum Mechanics (CQM) (Abramsky and Coecke 2004) and it becomes apparent that the string diagrams describing CQM are tensor networks endowed with a graphical language in the form of a rewrite-system (a.k.a. diagrammatic algebra with string diagrams). The language of string diagrams places syntactic structures and quantum processes on equal footing, and thus allows the canonical instantiation of grammar-aware quantum models for NLP.

In this work, we bring DisCoCat to the current age of noisy intermediate-scale quantum (NISQ) devices by performing the first-ever proof-of-concept QNLP experiment on actual quantum processors. We employ the framework introduced in Meichanetzidis et al. (2020) by adopting the paradigm of parameterised quantum circuits (PQCs) as quantum machine learning models (Schuld et al. 2020; Benedetti et al. 2019), which currently dominates near-term algorithms. PQCs can be used to parameterise quantum states and processes, as well as complex probability distributions, and so they can be used in NISQ machine learning pipelines. The framework we use in this work allows for the execution of experiments involving non-trivial text corpora, which moreover involves complex grammatical structures. The specific task we showcase here is binary classification of sentences in a supervised-learning hybrid classical-quantum QML setup.

2 The model

The DisCoCat model relies on algebraic models of grammar which use types and reduction-rules to mathematically formalise syntactic structures of sentences. Historically, the first formulation of DisCoCat employed pregroup grammars (Appendix A), which were developed by Lambek (2008). However, *any other typeological grammar*, such as Combinatory Categorial Grammar (CCG), can be used to instantiate DisCoCat models.

In this work, we will work with *pregroup grammars*, to stay close to the existing DisCoCat literature. In a pregroup grammar, a sentence's type is *composed of* a finite product of words $\sigma = \prod_i w_i$. A parser tags a word $w \in \sigma$ with its part of speech. Accordingly, w is assigned a *pregroup type* $t_w = \prod_i b_i^{\kappa_i}$ comprising a product of *basic (or atomic) types* b_i from the finite set B. Each type carries an *adjoint order* $\kappa_i \in \mathbb{Z}$. Pregroup parsing is efficient; specifically, it is linear time under reasonable assumptions (Preller 2007). The type of a sentence is the product of the types of its words and it is deemed grammatical if it type-reduces to the special type $s^0 \in B$, i.e. the sentence-type, $t_\sigma = \prod_w t_w \rightarrow s^0$. Reductions are performed by iteratively applying pairwise annihilations of basic types with adjoint orders of the form $b^i b^{i+1}$. As an example, consider the *grammatical reduction*:

$$\begin{aligned}
&t_{\text{Romeo who loves Juliet dies}} &=\\
&t_{\text{Romeo}}\, t_{\text{who}}\, t_{\text{loves}}\, t_{\text{Juliet}}\, t_{\text{dies}} &=\\
&(n^0)(n^1 n^0 s^{-1} n^0)(n^1 s^0 n^{-1})(n^0)(n^1 s^0) &\rightarrow\\
&\cancel{n^0 n^1} n^0 s^{-1} \cancel{n^0 n^1} s^0 \cancel{n^{-1} n^0} n^1 s^0 \;\rightarrow\; n^0 \cancel{s^{-1} s^0} n^1 s^0 &\rightarrow\\
&\cancel{n^0 n^1} s^0 \rightarrow s^0.
\end{aligned}$$

At the core of DisCoCat is a process-theoretic model of natural language meaning. Process theories are alternatively known as symmetric monoidal (or tensor) categories (Baez and Stay 2009). Process networks such as those that manifest in DisCoCat can be represented graphically with *string diagrams* (Selinger 2010). String diagrams are not just convenient graphical notation, but they constitute a formal graphical language for reasoning about complex process networks (see Appendix B for an introduction to string diagrams and concepts relevant to this work). String diagrams are generated by boxes with input and output wires, with each wire carrying a type. Boxes can be composed to form process networks by wiring outputs to inputs and making sure the types are respected. Output-only processes are called *states* and input-only processes are called *effects*.

A grammatical reduction is viewed as a process and so it can be represented as a string diagram. In string diagrams representing pregroup-grammar reductions, words are represented as states and pairwise type-reductions are represented by a pattern of nested cup-effects (wires bent in a U-shape), and identities (straight wires). Wires in the string diagram carry the label of the basic type being reduced. As an example, in Fig. 1 we show the string diagram representing the pregroup reductions for 'Romeo who loves Juliet dies'. Only the s-wire is left open, which is the *witness of grammaticality*.

Given a string diagram resulting from the grammatical reduction of a sentence, we can instantiate a model for natural language processing by giving *semantics* to the string diagram. This two-step process of constructing a model, where syntax and the semantics are treated separately, is the origin of the framework's name; 'Compositional' refers to the string diagram describing structure and 'Distributional' refers to the semantic spaces where meaning is encoded and processed. Any choice of semantics that respects the compositional structure is allowed, and is implemented by component-wise substitution of the boxes and wires in the diagrams. Such a structure preserving mapping constitutes a *functor*, and ensures that the model instantiation is 'canonical'. Valid choices of semantics range from neural networks, to tensor networks (which was the default choice for the

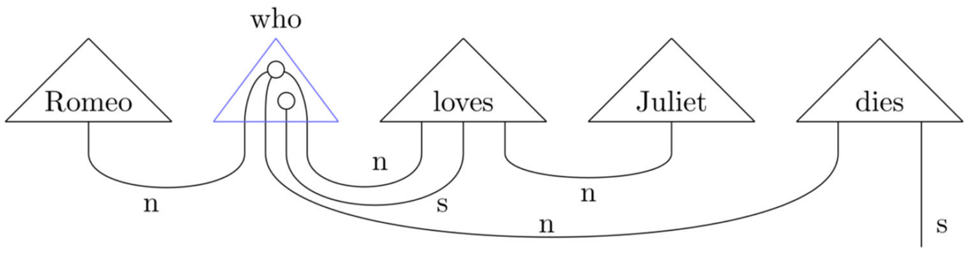

Fig. 1 Diagram for 'Romeo who loves Juliet dies'. The grammatical reduction is generated by the nested pattern of non-crossing cups, which connect words through wires of types n or s. Grammaticality is verified by only one s-wire left open. The diagram represents the meaning of the whole sentence from a process-theoretic point of view. The relative pronoun 'who' is modeled by the Kronecker tensor. Interpreting the diagram in CQM, it represents a quantum state

majority of the DisCoCat literature), to quantum processes, and even hybrid combinations involving components from the whole range of available choice by interpreting the string diagram correspondingly.

In this work, we focus on the latter choice of *quantum processes*, and in particular, we will be realising quantum processes using *pure quantum circuits*. As we have described in Meichanetzidis et al. (2020), the string diagram of the syntactic structure of a sentence σ can be canonically mapped to a PQC $C_\sigma(\theta_\sigma)$ over the parameter set θ. The key idea here is that such circuits *inherit their architecture*, in terms of a particular connectivity of entangling gates, from the *grammatical reduction* of the sentence.

Quantum circuits also, being part of pure quantum theory, enjoy a graphical language in terms of string diagrams. The mapping from sentence diagram to quantum circuit begins simply by reinterpreting a sentence diagram, such as that of Fig. 1, as a diagram in categorical quantum mechanics (CQM). The word-state of word w in a sentence diagram is mapped to a pure quantum state prepared from a trivial reference product-state by a PQC as $|w(\theta_w)\rangle = C_w(\theta_w)|0\rangle^{\otimes q_w}$, where $q_w = \sum_{b\in t_w} q_b$. The width of the circuit depends on the number of qubits assigned to each pregroup type $b \in B$ from which the word-types are composed and cups are mapped to Bell effects.

Given a sentence σ, we instantiate its quantum circuit by first concatenating in parallel the word-circuits of each word as they appear in the sentence, corresponding to performing a tensor product, $C_\sigma(\theta_\sigma) = \bigotimes_w C_w(\theta_w)$ which prepares the state $|\sigma(\theta_\sigma)\rangle$ from the all-zeros basis state. As such, a sentence is parameterised by the concatenation of parameters of its words, $\theta_\sigma = \cup_{w\in\sigma}\theta_w$. The parameters θ_w determine the word-embedding $|w(\theta_w)\rangle$. In other words, we use the Hilbert space as a feature space (Havlíček et al. 2019; Schuld and Killoran 2019; Lloyd et al. 2020) in which the word-embeddings are defined. Finally, we apply Bell effects as dictated by the cup pattern in the grammatical reduction, a function whose result we shall denote $g_\sigma(|\sigma(\theta_\sigma)\rangle)$. Note that in general this procedure prepares an unnormalised quantum state. In the special case where no qubits are assigned to the sentence type, i.e. $q_s = 0$, then it is an amplitude which we write as $\langle g_\sigma|\sigma(\theta_\sigma)\rangle$. Formally, this mapping constitutes a *parameterised functor* from the pregroup grammar category to the category of quantum circuits. The parameterisation is defined via a function from the set of parameters to functors from the aforementioned source and target categories.

Our model has hyperparameters (Appendix E). The wires of the DisCoCat diagrams we consider carry types n or s. The number of qubits that we assign to each pregroup type are q_n and q_s. These determine the *arity* of each word, i.e. the width of the quantum circuit that prepares each word-state. We set $q_s = 0$ throughout this work, which establishes that the sentence-circuits represent *scalars*. For a unary word w, i.e. a word-state on 1 qubit, we choose to prepare using two rotations as $R_z(\theta_w^2)R_x(\theta_w^1)|0\rangle$. For a word w of arity $k \geq 2$, we use a depth-d IQP-style parameterisation (Havlíček et al. 2019) consisting of d layers where each layer consists of a layer of Hadamard gates followed by a layer of controlled-Z rotations $CR_z(\theta_w^i)$, such that $i \in \{1, 2, \ldots, d(k-1)\}$. Such circuits are in part motivated by the conjecture that circuits involving them are classically hard to evaluate (Havlíček et al. 2019). The relative pronoun 'who' is mapped to the GHZ circuit, i.e. the circuit that prepares a GHZ state on the number of qubits as determined by q_n and q_s. This is justified by prior work where relative pronouns and other functional words are modelled by a Kronecker tensor (a.k.a. 'spider'), whose entries are all zeros except when all indices are the same for which case the entries are ones (Sadrzadeh et al. 2013; 2014). It is also known as a 'copy' tensor, as it copies the computational basis.

In Fig. 2, we show an example of choices of word-circuits for specific numbers of qubits assigned to each basic pregroup type (Appendix Fig. 8). In Fig. 3, we show the corresponding circuit to 'Romeo who loves Juliet dies'. In practice, we perform the mapping of sentence diagrams to quantum circuits using the Python library `DisCoPy` (de Felice et al. 2020), which provides a data structure for monoidal string-diagrams and enables the instantiation of functors, including functors based on PQCs.

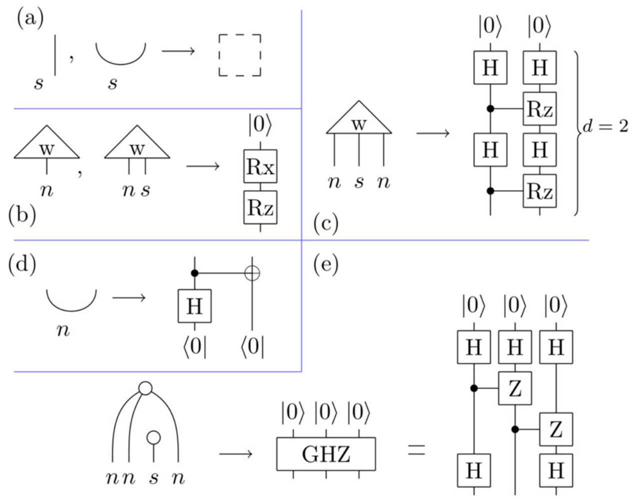

Fig. 2 Example instance of mapping from sentence diagrams to PQCs where $q_n = 1$ and $q_s = 0$. (a) The dashed square is the empty diagram. In this example, (b) unary word-states are prepared by parameterised R_x rotations followed by R_z rotations and (c) k-ary word-states are prepared by parameterised word-circuits of width k and depth $d = 2$. (d) The cup is mapped to a Bell effect, i.e. a CNOT followed by a Hadamard on the control and postselection on $\langle 00|$. (e) The Kronecker tensor modelling the relative pronoun is mapped to a GHZ state

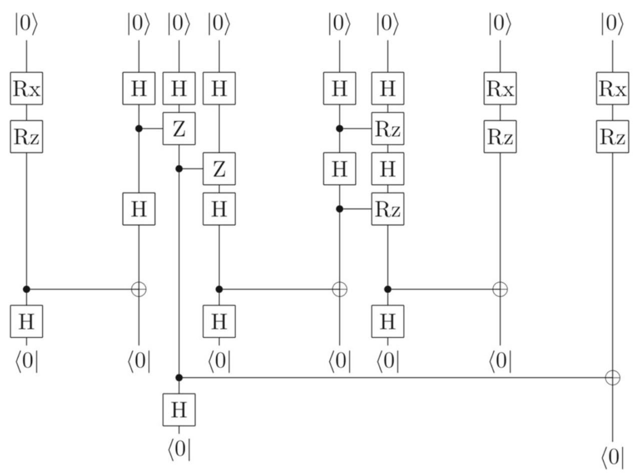

Fig. 3 The PQC to which ‘Romeo who loves Juliet dies’ of Fig. 1 is mapped, with the choices of hyper-parameters of Fig. 2. As $q_s = 0$, the circuit represents a scalar

Here, a motivating remark is in order. In ‘classical’ implementations of DisCoCat, where the semantics chosen in order to realise a model is in terms of tensor networks, a sentence diagram represents a vector which results from a tensor contraction. In this case, meanings of words are encoded in the state-tensors in terms of cooccurrence frequencies or other vector-space word-embeddings (Mikolov et al. 2013). In general, tensor contractions are exponentially expensive to compute. The cost scales exponentially with the order of the largest tensors present in the tensor network and the base of the scaling is the dimension of the vector spaces carried by the wires. However, tensor networks resulting from interpreting syntactic string diagrams over vector spaces and linear maps do not have a generic topology; rather they are tree-like. This means that contracting tensor networks whose connectivity is given by pregroup reductions are efficiently contractable as a function of the dimension of the wires carrying the vector spaces playing the role of semantic spaces. Even in this case however, the dimension of the wires for NLP-relevant applications can become prohibitively large (order of hunderds) in practice. In a fault-tolerant quantum computing setting, ideally, as is proposed in Zeng and Coecke (2016), one has access to a QRAM and one would be able to efficiently encode such tensor entries as quantum amplitudes using only $\lceil \log_2 d \rceil$ qubits to encode a d-dimensional vector. However, building a QRAM currently remains challenging (Aaronson 2015). In the NISQ case, we still attempt to take advantage of the tensor-product structure defined by a collection of qubits which provides an exponentially large Hilbert space as a function of the number of qubits, and can be used as a feature-space in which the word-embeddings can be trained. Consequently, we adopt the paradigm of QML in terms of PQCs to carry out near-term QNLP tasks. Any possible quantum advantage is to be identified heuristically on a case by case basis depending on the task, the data, and the available quantum computing resources.

3 Classification task

Now that we have established our construction of sentence circuits, we describe a simple QNLP task. The dataset or ‘labelled corpus’ $K = \{(D_\sigma, l_\sigma)\}_\sigma$ is a finite set of sentence-diagrams $\{D_\sigma\}_\sigma$ constructed from a finite vocabulary of words V. Each sentence has a binary label $l_\sigma \in \{0, 1\}$. In this work, the labels represent the sentences’ truth-values; 0 for False and 1 for True. Our setup trivially generalises to multi-class classification by assigning $q_s = \log_2(\#\text{classes})$ to the s-type wire and measuring in the Z-basis.

We split K into the training set Δ containing the first $\lfloor p|\{D_\sigma\}_\sigma| \rceil$ of the sentences, where $p \in (0, 1)$, and the test set E containing the rest.

We define the *predicted label* as

$$l_\sigma^{\text{pr}}(\theta_\sigma) = |\langle g_\sigma | \sigma(\theta_\sigma) \rangle|^2 \in [0, 1] \tag{1}$$

from which we can obtain the binary label by rounding to the nearest integer $\lfloor l_\sigma^{\text{pr}} \rceil \in \{0, 1\}$.

The parameters of the words need to be optimised (or trained) so that the predicted labels match the labels in the training set. The optimiser we invoke is SPSA (Spall 1998), a gradient-free optimiser which has shown adequate performance in noisy settings (Bonet-Monroig et al. 2021) (Appendix F). The *cost function* we define is

$$L(\theta) = \sum_{\sigma \in \Delta} (l_\sigma^{\text{pr}}(\theta_\sigma) - l_\sigma)^2. \tag{2}$$

Minimising the cost function returns the optimal parameters $\theta^* = \text{argmin} L(\theta)$ from which the model predicts the labels $l_\sigma^{\text{pr}}(\theta^*)$. Essentially, this constitutes *learning a functor* from the grammar category to the category of quantum circuits. We then quantify the performance by the training and test errors e_Δ and e_E, as the proportion of labels predicted incorrectly:

$$e_A = \frac{1}{|A|} \sum_{\sigma \in A} \left| \lfloor l_\sigma^{\text{pr}}(\theta^*) \rceil - l_\sigma \right| , \quad A = \Delta, E.$$

This supervised learning task of binary classification for sentences is a special case of *question answering* (QA) (de

Felice et al. 2020; Chen et al. 2020; Zhao et al. 2020); questions are posed as statements and the truth labels are the binary answers. After training on Δ, the model predicts the answer to a previously unseen question from E, which comprises sentences containing words all of which have appeared in Δ. The optimisation is performed over the parameters of all the sentences in the training set $\theta = \cup_{\sigma\in\Delta}\theta_\sigma$. In our experiments, each word appears at least once in the training set and so $\theta = \cup_{w\in V}\theta_w$. Note that what is being learned are the inputs, i.e. the quantum word embeddings, to an entangling process corresponding to the grammar. Recall that a given sentence-circuit does not necessarily involve the parameters of every word. However, every word appears in at least one sentence, which introduces classical correlations between the sentences. This makes such a learning task possible.

In this work, we use artificial data in the form of a very small-scale corpus of grammatical sentences. We randomly generate sentences using a simple context-free grammar (CFG) whose production rules we define. Each sentence then is accompanied by its syntax tree by definition of our generation procedure. The syntax tree can be cast in string-diagram form, and each CFG-generated sentence-diagram can then be transformed into a DisCoCat diagram (see Appendix C for details). Even though the data is synthetic, we curate the data by assigning labels by hand so that the truth values among the sentences are consistent with a story, rendering the classification task semantically non-trivial.

Were one to use real-world datasets of labelled-sentences, one could use a *parser* in order to obtain the syntactic structures of the sentences; the rest of our pipeline would still remain. Since the writing of this manuscript, the open-source Python package `lambeq` (Kartsaklis et al. 2021) has been made available, which couples to the end-to-end parser `Bobcat`. The parser, given a sentence, returns its syntax tree as a grammatical reduction in the Combinatory Categorial Grammar (CCG). The package also has the capability to translate CCG reductions to pregroup reductions (Yeung and Kartsaklis 2021), and so effectively it is the first practical tool introduced for large-scale parsing in pregroup grammar. The string diagrams in `lambeq` are encoded using `DisCoPy`, and thus enables the instantiation of large-scale DisCoCat models on real-world data.

3.1 Classical simulation

We first show results from classical simulations of the QA task. The sentence circuits are evaluated exactly on a classical computer to compute the predicted labels in Eq.1. We consider the corpus K_{30} of 30 sentences sampled from the vocabulary of 7 words (Appendix D) and we set $p = 0.5$. In Fig. 4 we show the convergence of the cost function, for $q_n = 1$ and $q_n = 2$, for increasing word-circuit depth d. To clearly show the decrease in training and test errors as a function of d when invoking the global optimiser `basinhopping` (Appendix F).

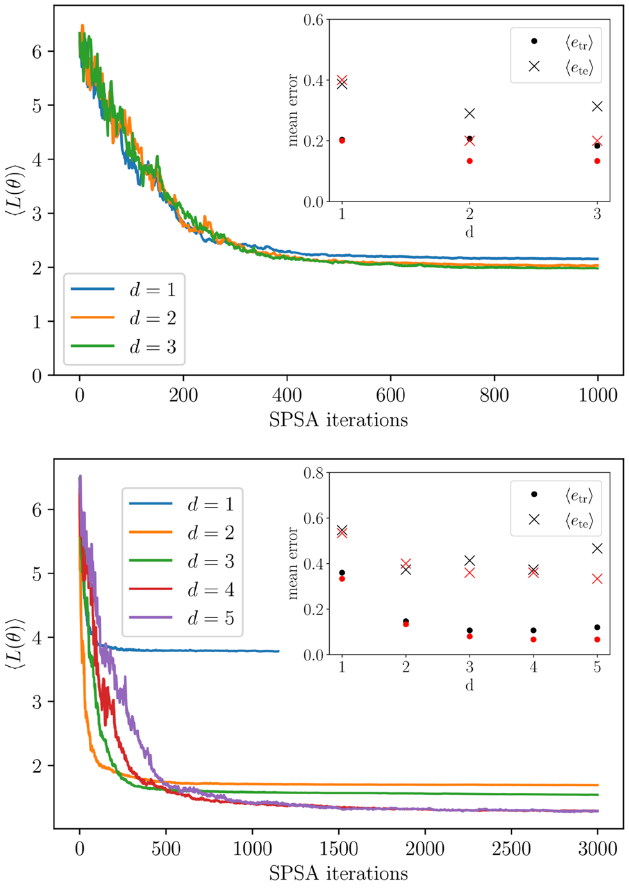

Fig. 4 Convergence of mean cost function $\langle L(\theta)\rangle$ vs number of `SPSA` iterations for corpus K_{30}. A lower minimum is reached for larger d. (Top) $q_n = 1$ and $|\theta| = 8 + 2d$. Results are averaged over 20 realisations. (Bottom) $q_n = 2$ and $|\theta| = 10d$. Results are averaged over 5 realisations. (Insets) Mean training and test errors $\langle e_{\text{tr}}\rangle$, $\langle e_{\text{te}}\rangle$ vs d. Using the global optimisation `basinhopping` with local optimisation `Nelder-Mead` (red), the errors decrease with d

3.2 Experiments on IBMQ

We now turn to readily available NISQ devices provided by the IBMQ in order to estimate the predicted labels in Eq.1.

Before each circuit can be run on a backend, in this case a superconducting quantum processor, it first needs to be compiled. A quantum compiler takes as input a circuit and a backend and outputs an equivalent circuit which is compatible with the backend's topology. A quantum compiler also aims to minimise the most noisy operations. For IBMQ, the gate most prone to erros is the entangling CNOT gates. The compiler we use in this work is TKET (Sivarajah et al. 2020), and for each circuit-run on a backend, we use the maximum allowed number of shots (Appendix G).

We consider the corpus K_{16} from 6 words (Appendix D) and set $p = 0.5$. For every evaluation of the cost function under optimisation, the circuits were run on the IBMQ quantum computers `ibmq_montreal` and `ibmq_toronto`. In Fig. 5, we show the convergence of the cost function under `SPSA` optimisation and report the training and testing errors for different choices of hyperparameters. This constitutes the first non-trivial QNLP experiment on a programmable quantum processor. According to Fig. 4, scaling up the word-circuits results in improvement in training and testing errors, and remarkably, we observe this on the quantum computer, as well. This is important for the scalability of our experiment when future hardware allows for greater circuit sizes and thus richer quantum-enhanced feature spaces and grammatically more complex sentences.

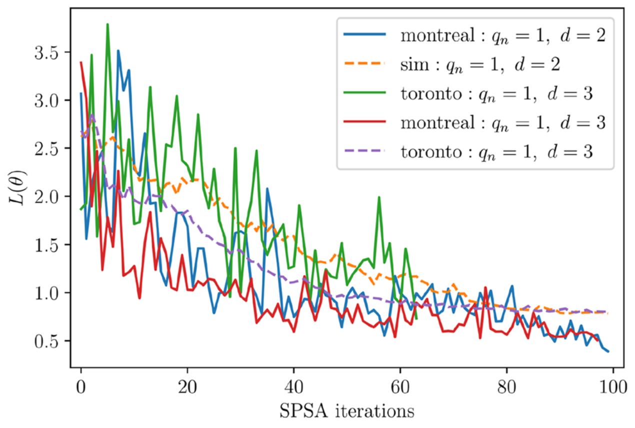

Fig. 5 Convergence of the cost $L(\theta)$ evaluated on quantum computers vs `SPSA` iterations for corpus K_{16}. For $q_n = 1$, $d = 2$, for which $|\theta| = 10$, on `ibmq_montreal` (blue) we obtain $e_{\text{tr}} = 0.125$ and $e_{\text{te}} = 0.5$. For $q_n = 1$, $d = 3$, where $|\theta| = 13$, on `ibmq_toronto` (green) we get $e_{\text{tr}} = 0.125$ and a lower testing error $e_{\text{te}} = 0.375$. On `ibmq_montreal` (red) we get both lower training and testing errors, $e_{\text{tr}} = 0$, $e_{\text{te}} = 0.375$ than for $d = 2$. In all cases, the CNOT-depth of any sentence-circuit after TKET-compilation is at most 3. Classical simulations (dashed), averaged over 20 realisations, agree with behaviour on IBMQ for both cases $d = 2$ (yellow) and $d = 3$ (purple)

4 Discussion and outlook

We have performed the first-ever quantum natural language processing experiment by means of classification of sentences annotated with binary labels, a special case of QA, on actual quantum hardware. We used a compositional-distributional model of meaning, DisCoCat, constructed by a structure-preserving mapping from grammatical reductions of sentences to PQCs. This proof-of-concept work serves as a demonstration that QNLP is possible on currently available quantum devices and that it is a line of research worth exploring.

A remark on postselection is in order. QML-based QNLP tasks such as the one implemented in this work rely on the optimisation of a scalar cost function. In general, estimating a scalar encoded in an amplitude on a quantum computer requires either postselection or coherent control over arbitrary circuits so that a swap test or a Hadamard test can be performed (Appendix H). Notably, in special cases of interest to QML, the Hadamard test can be adapted to NISQ technologies (Mitarai and Fujii 2019; Benedetti et al. 2020). In its general form, however, the depth-cost resulting after compilation of controlled circuits becomes prohibitable with current quantum devices. However, given the rapid improvement in quantum computing hardware, we envision that such operations will be within reach in the near-term.

Future work includes experimentation with other grammars, such as CCG which returns tree-like diagrams, and using them to construct PQC-based functors, as is done in Socher et al. (2013) but with neural networks. This for example would enable the design of PQC-based functors that do not require postselection, such as the pregroup-based models where in order for each Bell effect to take place one needs to postselect on measurements involving the qubits on which one wishes to realise a Bell effect.

We also look toward more complex QNLP tasks such as sentence similarity and work with real-world large-scale data using a pregroup parser, as made possible with `lambeq` (Kartsaklis et al. 2021). In that context, regularisation techniques during training will become important, which is an increasingly relevant topic for QML that in general deserves more attention (Benedetti et al. 2019).

In addition, our DisCoCat-based QNLP framework is naturally generalisable to accommodate mapping sentences to quantum circuits involving mixed states and quantum channels. This is useful as mixed states allow for modelling lexical entailment and ambiguity (Piedeleu et al. 2015; Bankova et al. 2016). As also stated above, it is possible to define functors in terms of hybrid models where both neural networks and PQCs are involved, where heuristically one aims to quantify the possible advantage of such models compared to strictly classical ones.

Furthermore, note that the word-embeddings are learned in-task in this work. However, training PQCs to prepare quantum states that serve as word embeddings can be achieved by using the usual NLP objectives (Mikolov et al. 2013). It is interesting to verify that such pretrained word embeddings can be useful in downstream tasks, such as the simple classification task presented in this work.

Finally, looking beyond the DisCoCat model, it is well-motivated to adopt the recently introduced DisCoCirc model (Coecke 2020) of meaning and its mapping to PQCs (Coecke et al. 2020), which allows for QNLP experiments on text-scale real-world data in a fully-compositional framework. In this model, nouns are treated as first-class citizen 'entities' of a text and makes *sentence composition* explicit. Entities go through gates which act as modifiers on them, modelling for example the application of adjectives or verbs. The model also considers higher-order modifiers, such as adverbs modifying verbs. This interaction structure, viewed as a process network, can again be used to instantiate models in terms of neural networks, tensor networks, or quantum circuits. In the latter case, entities are modelled as density matrices carried by wires and their modifiers as quantum channels.

Appendix

In this supplementary material, we begin by briefly reviewing pregroup grammar. We then provide the necessary background to the graphical language of process theories describe our procedure for generating random sentence diagrams using a context-free grammar. For completeness we include the three labelled corpora of sentences we used in this work. Furthermore, we show details of our mapping from sentence diagrams to quantum circuits. Finally, we give details on the optimisation methods we used for our supervised quantum machine learning task and the specific compilation pass we used from CQC's compiler, TKET.

Appendix A: Pregroup grammar

Pregroup grammars where introduced by Lambek as an algebraic model for grammar (Lambek 2008).

A pregroup grammar G is freely generated by the basic types in a finite set $b \in B$. Basic types are decorated by an integer $k \in \mathbb{Z}$, which signifies their adjoint order. Negative integers $-k$, with $k \in \mathbb{N}$, are called *left adjoints* of order k and positive integers $k \in \mathbb{N}$ are called *right adjoints*. We shall refer to a basic type to some adjoint order (include the zeroth order) simply as '*type*'. The zeroth order $k = 0$ signifies no adjoint action on the basic type and so we often omit it in notation, $b^0 = b$.

The pregroup algebra is such that the two kinds of adjoint (left and right) act as left and right inverses under multiplication of basic types

$$b^k b^{k+1} \to \epsilon \to b^{k+1} b^k,$$

where $\epsilon \in B$ is the trivial or *unit* type. The left hand side of this reduction is called a *contraction* and the right hand side an *expansion*. Pregroup grammar also accommodates *induced steps* $a \to b$ for $a, b \in B$. The symbol '$\to$' is to be read as 'type-reduction' and the pregroup grammar sets the rules for which reductions are valid.

Now, to go from word to sentence, we consider a finite set of words called the *vocabulary* V. We call the *dictionary* (or lexicon) the finite set of entries $D \subseteq V \times (B \times \mathbb{Z})^*$. The star symbol A^* denotes the set of finite strings that can be generated by the elements of the set A. Each dictionary entry assigns a *product* (or string) of types to a word $t_w = \prod_i b_i^{k_i}, k_i \in \mathbb{Z}$.

Finally, a pregroup grammar G generates a language $L_G \subseteq V^*$ as follows. A sentence is a sequence (or list) of words $\sigma \in V^*$. The type of a sentence is the product of types of its words $t_\sigma = \prod_i t_{w_i}$, where $w_i \in V$ and $i \leq |\sigma|$. A sentence is *grammatical*, i.e. it belongs to the language generated by the grammar $\sigma \in L_G$, if and only if there exists a sequence of reductions so that the type of the sentence reduces to the special *sentence-type* $s \in B$ as $t_\sigma \to \cdots \to s$. Note that it is in fact possible to type-reduce grammatical sentences *only using contractions*.

Appendix B: String diagrams

String diagrams describing process theories are generated by states, effects, and processes. In Fig. 6, we comprehensively show these generators along with constraining equations on them. String diagrams for process theories formally describe process networks where only connectivity matters, i.e. which outputs are connected to which inputs. In other words, the length of the wires carries no meaning and the wires are freely deformable as long as the topology of the network is respected.

It is beyond the purposes of this work to provide a comprehensive exposition on diagrammatic languages. We

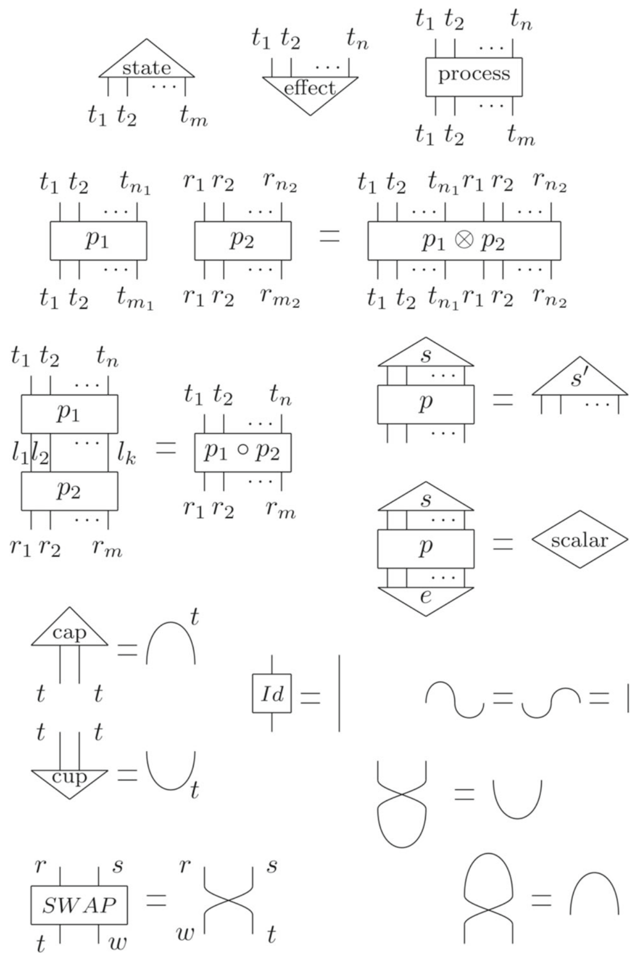

Fig. 6 Diagrams are read from top to bottom. States have only outputs, effects have only inputs, processes (boxes) have both input and output wires. All wires carry types. Placing boxes side by side is allowed by the monoidal structure and signifies parallel processes. Sequential process composition is represented by composing outputs of a box with inputs of another box. A process transforms a state into a new state. There are special kinds of states called caps and effects called cups, which satisfy the snake equation which relates them to the identity wire (trivial process). Process networks freely generated by these generators need not be planar, and so there exists a special process that swaps wires and acts trivially on caps and cups

provide the necessary elements which are used for the implementation of our QNLP experiments.

Appendix C: Random sentence generation with CFG

A context-free grammar generates a language from a set of production (or rewrite) rules applied on symbols. Symbols belong to a finite set Σ and There is a special type $S \in \Sigma$ called initial. Production rules belong to a finite set R and are of the form $T \rightarrow \prod_i T_i$, where $T, T_i \in \Sigma$. The application of a production rule results in substituting a symbol with a product (or string) of symbols. Randomly generating a sentence amounts to starting from S and randomly applying production rules uniformly sampled from the set R. The production ends when all types produced are terminal types, which are non other than words in the finite vocabulary V.

From a process theory point of view, we represent symbols as types carried by wires. Production rules are represented as boxes with input and output wires labelled by the appropriate types. The process network (or string diagram) describing the production of a sentence ends with a production rule whose output is the S-type. Then we randomly pick boxes and compose them backwards, always respecting type-matching when inputs of production rules are fed into outputs of other production rules. The generation terminates when production rules are applied which have no inputs (i.e. they are states), and they correspond to the words in the finite vocabulary.

In Fig. 7 (on the left hand side of the arrows), we show the string-diagram generators we use to randomly produce sentences from a vocabulary of words composed of nouns, transitive verbs, intransitive verbs, and relative pronouns. The corresponding types of these parts of speech are $N, TV, IV, RPRON$. The vocabulary is the union of the words of each type, $V = V_N \cup V_{TV} \cup V_{IV} \cup V_{RPRON}$.

Having randomly generated a sentence from the CFG, its string diagram can be translated into a pregroup sentence diagram. To do so, we use the translation rules shown in Fig. 7. Note that a cup labeled by the basic type b is used to represent a contraction $b^k b^{k+1} \rightarrow \epsilon$. Pregroup grammars

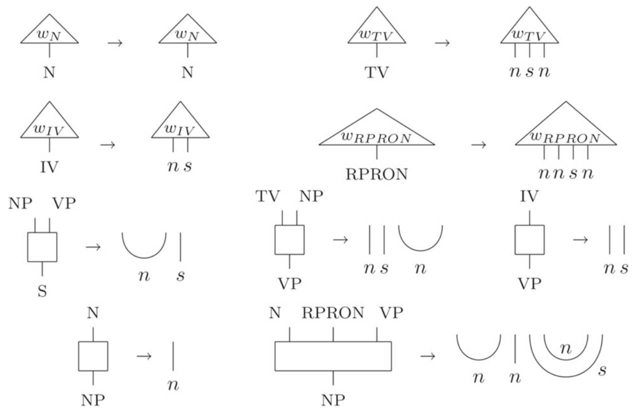

Fig. 7 CFG generation rules used to produce the corpora K_{30}, K_6, K_{16} used in this work, represented as string-diagram generators, where $w_N \in V_N$, $w_{TV} \in V_{TV}$, $w_{IV} \in V_{IV}$, $w_{RPRON} \in V_{RPRON}$. They are mapped to pregroup reductions by mapping CFG symbols to pregroup types, and so CFG-states are mapped to DisCoCat word-states and production boxes are mapped to products of cups and identities. Note that the pregroup unit ϵ is the empty wire and so it is never drawn. Pregroup type contractions correspond to cups and expansions to caps. Since grammatical reduction are achievable only with contractions, only cups are required for the construction of sentence diagrams

are weakly equivalent to context-free grammars, in the sense that they generate the same language (Buszkowski and Moroz 2007; Pentus 1993).

Appendix D: Corpora

Here, we present the sentences and their labels used in the experiments presented in the main text.

The types assigned to the words of this sentence are as follows. Nouns get typed as $t_{w \in V_N} = n^0$, transitive verbs are given type $t_{w \in V_{TV}} = n^1 s^0 n^{-1}$, intransitive verbs are typed $t_{w \in IV} = n^1 s^0$, and the relative pronoun is typed $t_{\text{who}} = n^1 n^0 s^{-1} n^0$.

Corpus K_{30} of 30 labeled sentences from the vocabulary $V_N = \{\text{'Dude'}, \text{'Walter'}\}$, $V_{TV} = \{\text{'loves'}, \text{'annoys'}\}$, $V_{IV} = \{\text{'abides'},\text{'bowls'}\}$, $V_{RPRON} = \{\text{'who'}\}$:
[('Dude who loves Walter bowls', 1),
('Dude bowls', 1),
('Dude annoys Walter', 0),
('Walter who abides bowls', 0),
('Walter loves Walter', 1),
('Walter annoys Dude', 1),
('Walter bowls', 1),
('Walter abides', 0),
('Dude loves Walter', 1),
('Dude who bowls abides', 1),
('Walter who bowls annoys Dude', 1),
('Dude who bowls bowls', 1),
('Dude who abides abides', 1),
('Dude annoys Dude who bowls', 0),
('Walter annoys Walter', 0),
('Dude who abides bowls', 1),
('Walter who abides loves Walter', 0),
('Walter who bowls bowls', 1),
('Walter loves Walter who abides', 0),
('Walter annoys Walter who bowls', 0),
('Dude abides', 1),
('Dude loves Walter who bowls', 1),
('Walter who loves Dude bowls', 1),
('Dude loves Dude who abides', 1),
('Walter who abides loves Dude', 0),
('Dude annoys Dude', 0),
('Walter who annoys Dude bowls', 1),
('Walter who annoys Dude abides', 0),
('Walter loves Dude', 1),
('Dude who bowls loves Walter', 1)]

Corpus K_6 of 6 labeled sentences from the vocabulary $V_N = \{\text{'Romeo'}, \text{'Juliet'}\}$, $V_{TV} = \{\text{'loves'}\}$, $V_{IV} = \{\text{'dies'}\}$, $V_{RPRON} = \{\text{'who'}\}$:
[('Romeo dies', 1.0),
('Romeo loves Juliet', 0.0),
('Juliet who dies dies', 1.0),
('Romeo loves Romeo', 0.0),
('Juliet loves Romeo', 0.0),
('Juliet dies', 1.0)]

Corpus K_{16} of 16 labeled sentences from the vocabulary $V_N = \{\text{'Romeo'}, \text{'Juliet'}\}$, $V_{TV} = \{\text{'loves'}, \text{'kills'}\}$, $V_{IV} = \{\text{'dies'}\}$, $V_{RPRON} = \{\text{'who'}\}$:
[('Juliet kills Romeo who dies', 0),
('Juliet dies', 1),
('Romeo who loves Juliet dies', 1),
('Romeo dies', 1),
('Juliet who dies dies', 1),
('Romeo loves Juliet', 1),
('Juliet who dies loves Juliet', 0),
('Romeo kills Juliet who dies', 0),
('Romeo who kills Romeo dies', 1),
('Romeo who dies dies', 1),
('Romeo who loves Romeo dies', 0),
('Romeo kills Juliet', 0),
('Romeo who dies kills Romeo', 1),
('Juliet who dies kills Romeo', 0),
('Romeo loves Romeo', 0),
('Romeo who dies kills Juliet', 0)]

Appendix E: Sentence to circuit mapping

Quantum theory has formally been shown to be a process theory. Therefore it enjoys a diagrammatic language in terms of string diagrams. Specifically, in the context of the quantum circuits we construct in our experiments, we use pure quantum theory. In the case of pure quantum theory, processes are unitary operations, or quantum gates in the context of circuits. The monoidal structure allowing for parallel processes is instantiated by the tensor product and sequential composition is instantiated by sequential composition of quantum gates.

In Fig. 8, we show the generic construction of the mapping from sentence diagrams to parameterised quantum circuits for the hyperparameters and parameterised word-circuits we use in this work.

A wire carrying basic pregroup type b is given q_b qubits. A word-state with only one output wire becomes a one-qubit-state prepared from $|0\rangle$. For the preparation of such unary states we choose the sequence of gates defining an Euler decomposition of one-qubit unitaries $R_z(\theta_1) \circ R_x(\theta_2) \circ R_z(\theta_3)$. Word-states with more than one output wires become multiqubit states on $k > 1$ qubits prepared by an IQP-style circuit from $\prod_{i=1}^{k} |0\rangle$. Such a word-circuit is composed of d-many layers. Each layer is composed of a layer of Hadamard gates followed by a layer in which every neighbouring pair of qubit wires is connected by a $CR_z(\theta)$

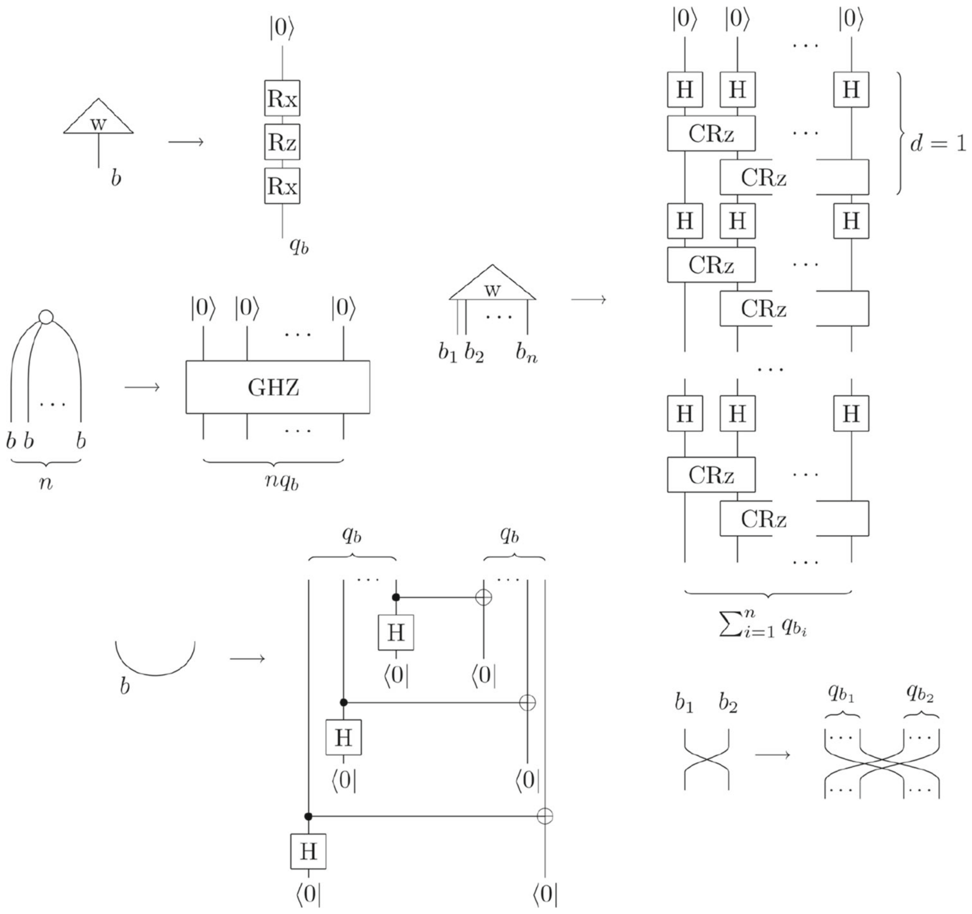

Fig. 8 Mapping from sentence diagrams to parameterised quantum circuits. Here, we show how the generators of sentence diagrams are mapped to generators of circuits, for the hyperparameters we consider in this work

gate, $\left(\otimes_{i=1}^{k} H\right) \circ \left(\otimes_{i=1}^{k-1} CR_z(\theta_i)_{i,i+1}\right)$. Since all CR_z gates commute with each other it is justified to consider this as a single layer, at least abstractly. The Kronecker tensor with n-many output wires of type b is mapped to a GHZ state on nq_b qubits. Specifically, GHZ is a circuit that prepares the state $\sum_{x=0}^{2^{q_b}} \bigotimes_{i=1}^{n} |\text{bin}(x)\rangle$, where bin is the binary expression of an integer. The cup of pregroup type b is mapped to q_b-many nested Bell effects, each of which is implemented as a CNOT followed by a Hadamard gate on the control qubit and postselection on $\langle 00|$.

Appendix F: Optimisation method

The gradient-free otpimisation method we use, Simultaneous Perturbation Stochastic Approximation (SPSA), works as follows. Start from a random point in parameter space. At every iteration pick randomly a direction and *estimate* the derivative by finite difference with step-size depending on c towards that direction. This requires two cost function evaluations. This provides a significant speed up the evaluation of $L(\theta)$. Then take a step of size depending on a towards (opposite) that direction if the derivative has negative (positive) sign. In our experiments we use `minimizeSPSA` from the Python package `noisyopt` https://github.com/andim/noisyopt, and we set $a = 0.1$ and $c = 0.1$, except for the experiment on `ibmq` for $d = 3$ for which we set $a = 0.05$ and $c = 0.05$.

Note that for classical simulations, we use just-in-time compilation of the cost function by invoking `jit` from `jax` (Bradbury et al. 2018). In addition, the choice of the squares-of-differences cost we defined in Eq.2 is not unique. One can as well use the binary cross entropy

$$L^{\text{BCE}}(\theta) = -\frac{1}{|\Delta|}\sum_{\sigma\in\Delta} l_\sigma \log l_\sigma^{\text{pr}}(\theta_\sigma) + (1-l_\sigma)\log(1-l_\sigma^{\text{pr}}(\theta_\sigma))$$

and the cost function can be minimised as well, as shown in Fig. 9.

In our classical simulation of the experiment, we also used `basinhopping` (Olson et al. 2012) in combination with `Nelder-Mead` (Gao and Han 2010) from the Python package `SciPy` https://pypi.org/project/scipy. `Nelder-Mead` is a gradient-free local optimisation method. `basinhopping` hops (or jumps) between basins (or local minima) and then returns the minimum over local minima of the cost function, where each minimum is found by `Nelder-Mead`. The hop direction is random. The hop is accepted according to a Metropolis criterion depending

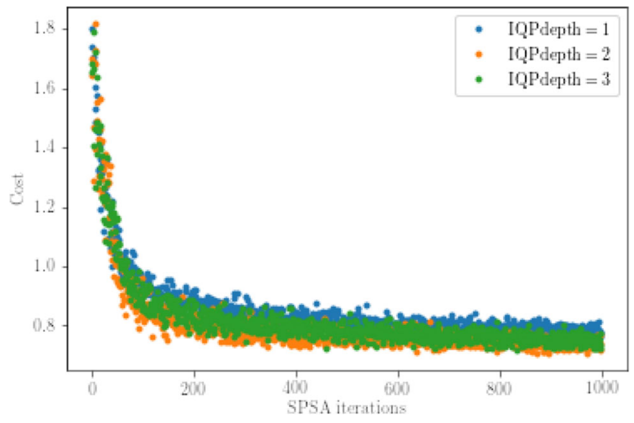

Fig. 9 Minimisation of binary cross entropy cost function L^{BCE} with `SPSA` for the question answering task for corpus K_{30}

on the the cost function to be minimised and a temperature. We used the default temperature value (1) and the default number of basin hops (100).

F.1: Error decay

In Fig. 10, we show the decay of mean training and test errors for the question answering task for corpus K_{30} simulated classically, which is shown as inset in Fig. 4. Plotting in log-log scale we reveal, at least initially, an algebraic decay of the errors with the depth of the word-circuits.

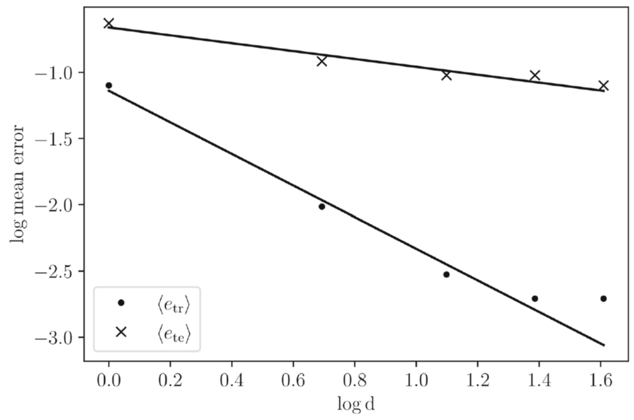

Fig. 10 Algebraic decay of mean training and testing error for the data displayed in Fig. 4 (bottom) obtained by `basinhopping`. Increasing the depth of the word-circuits results in algebraic decay of the mean training and testing errors. The slopes are $\log e_{\mathrm{tr}} \sim \log -1.2d$ and $\log e_{\mathrm{te}} \sim -0.3 \log d$. We attribute the existence of the plateau for e_{tr} at large depths is due the small scale of our experiment and the small values for our hyperparameters determining the size of the quantum-enhanced feature space

F.2: On the influence of noise to the cost function landscape

Regarding optimisation on a quantum computer, we comment on the effect of noise on the optimal parameters. Consider a successful optimisation of $L(\theta)$ performed on a NISQ device, returning θ^*_{NISQ}. However, if we instantiate the circuits $C_\sigma(\theta^*_{\mathrm{NISQ}})$ and evaluate them on a *classical computer* to obtain the predicted labels $l^{\mathrm{CC}}_{\mathrm{pr}}(\theta^*_{\mathrm{NISQ}})$, we observe that these can in general differ from the labels $l^{\mathrm{NISQ}}_{\mathrm{pr}}(\theta^*_{\mathrm{NISQ}})$ predicted by evaluating the circuits on the *quantum computer*. In the context of a fault-tolerant quantum computer, this should not be the case. However, since there is a non-trivial coherent-noise channel that our circuits undergo, it is expected that the optimiser's result are affected in this way.

Appendix G: Quantum compilation

In order to perform quantum compilation, we use `pytket` (Sivarajah et al. 2020). It is a Python module for interfacing with CQC's TKET, a toolset for quantum programming. From this toolbox, we need to make use of compilation passes.

At a high level, quantum compilation can be described as follows. Given a circuit and a device, quantum operations are decomposed in terms of the devices native gateset. Furthermore, the quantum circuit is reshaped in order to make it compatible with the device's topology (Cowtan et al. 2019). Specifically, the compilation pass that we use is `default_compilation_pass(2)`. The integer option is set to 2 for maximum optimisation under compilation https://github.com/CQCL/pytket.

Circuits written in `pytket` can be run on other devices by simply changing the backend being called, regardless whether the hardware might be fundamentally different in terms of what physical systems are used as qubits. This makes TKET it platform agnostic. We stress that on IBMQ machines specifically, the native gates are arbitrary single-qubit unitaries ('U3' gate) and entangling controlled-not gates ('CNOT' or 'CX'). Importantly, CNOT gates show error rates which are one or even two orders of magnitude larger than error rates of U3 gates. Therefore, we measure the depth of or circuits in terms of the CNOT-depth. Using `pytket` this can be obtained by invoking the command `depth_by_type(OpType.CX)`.

For both backends used in this work, `ibmq_montreal` and `ibmq_toronto`, the reported quantum volume is 32 and the maximum allowed number of shots is 2^{13}.

Appendix H: Swap test and Hadamard test

In our binary classification NLP task, the predicted label is the norm squared of zero-to-zero transition amplitude where the unitary U represents the word-circuits and the circuits that implement the Bell effects as dictated by the grammatical structure. Estimating $|\langle 0\ldots 0|U|0\ldots 0\rangle|^2$, or the amplitude $\langle 0\ldots 0|U|0\ldots 0\rangle$ itself in case one wants to define a cost function where it appears instead of its norm, can be done by postselecting on $\langle 0\ldots 0|$. However, postselection costs exponential time in the number of postselected qubits; in our case, it needs to discard all bitstring sampled from the quantum computer that have Hamming weight other than zero. This is the procedure we follow in this proof of concept experiment, as we can afford doing so due to the small circuit sizes.

In such a setting, postselection can be avoided by using the swap test to estimate the normed square of the amplitude or the Hadamard test for the amplitude itself (Aharonov et al. 2008). See Fig. 11 for the corresponding circuits of those routines. In Fig. 12, we show how the swap test or the Hadamard test can be used to estimate the amplitude represented by the postselected sentence-circuit of Fig. 3. Furthermore, at least for tasks that are defined such that every sentence corresponds to a circuit, we argue that sentences do not grow arbitrarily long, and so the cost of evaluating the cost function is upper bounded in practical applications.

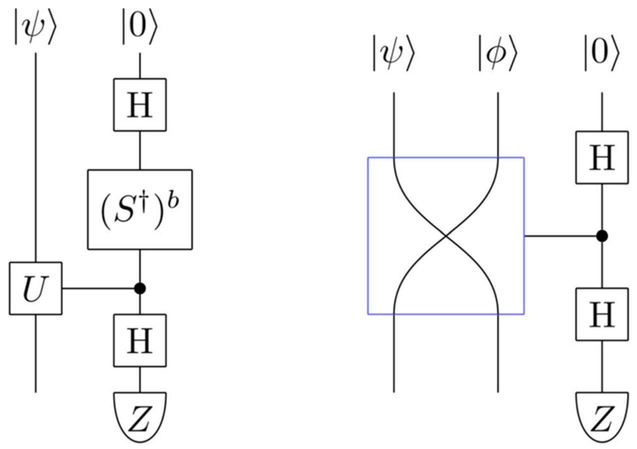

Fig. 11 (Left) Circuit for the Hadamard test. Measuring the control qubit in the computational basis allows one to estimate $\langle Z\rangle = \mathrm{Re}(\langle\psi|U|\psi\rangle)$ if $b = 0$, and $\langle Z\rangle = \mathrm{Im}(\langle\psi|U|\psi\rangle)$ if $b = 1$. The state ψ can be a multiqubit state, and in this work we are interested in the case $\psi = |0\ldots 0\rangle$. (Right) Circuit for the swap test. Sampling from the control qubit allows one to estimate $\langle Z\rangle = |\langle\psi|\phi\rangle|^2$

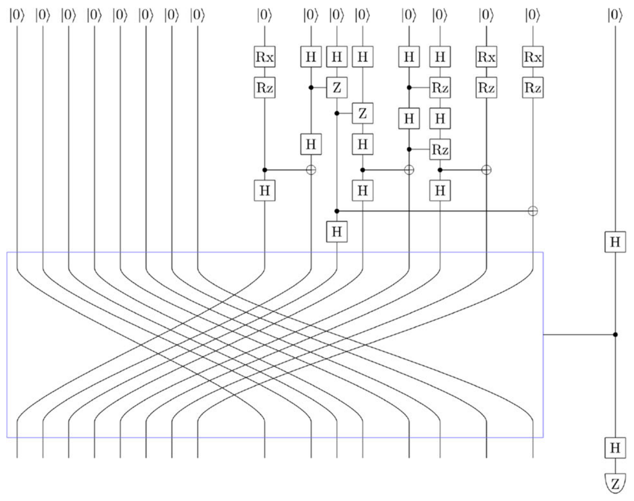
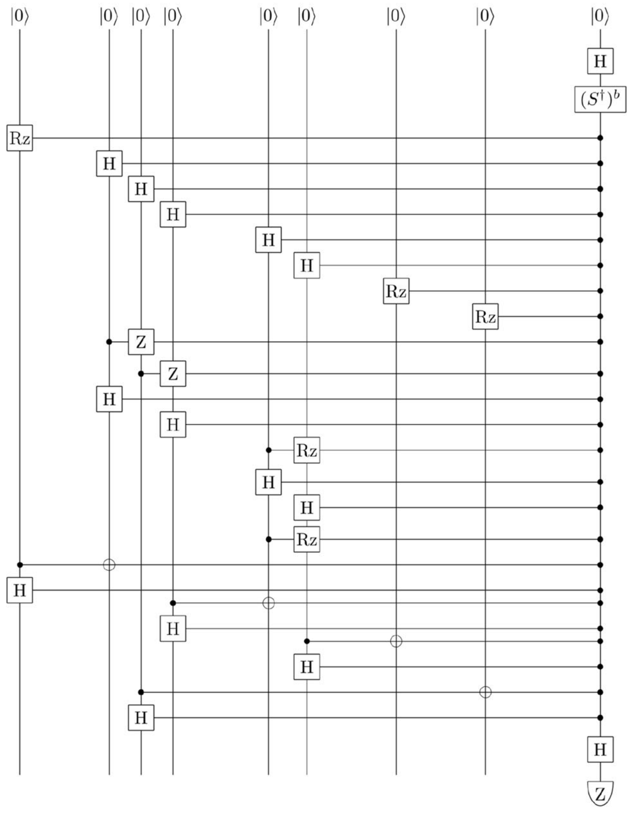

Fig. 12 Use of swap test (top) and Hadamard test (bottom) to estimate the norm-squared of the amplitude or the amplitude itself respectively, which is represented by the postselected circuit of Fig. 3

Acknowledgements KM thanks Vojtech Havlicek and Christopher Self for discussions on QML, Robin Lorenz and Marcello Benedetti for comments on the manuscript, and the TKET team at Quantinuum for technical advice on quantum compilation on IBMQ machines. We acknowledge the use of IBM Quantum services for this work.

Author contribution All authors contributed to the theory, the design of the model, and the high-level definition of the classification task. AT and GDF wrote the DisCoPy library (de Felice et al. 2020) and tested early versions of the experiment on `ibmq_singapore`. KM generated and annotated the data, implemented the simulations and experiments, and wrote the manuscript. BC led the project.

Funding KM is grateful to the Royal Commission for the Exhibition of 1851 for financial support under a Postdoctoral Research Fellowship. AT thanks Simon Harrison for financial support through the Wolfson Harrison UK Research Council Quantum Foundation Scholarship.

Data availability The toy datasets of generated sentences used in this work can be found in Appendix D. Further information about the datasets generated and analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare no competing interests.

Disclaimer The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

References

Aaronson S (2015) Read the fine print. Nat Phys 11(4):291–293

Abramsky S, Coecke B (2004) A categorical semantics of quantum protocols. In: Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004., pp. 415–425. https://doi.org/10.1109/LICS.2004.1319636

Aharonov D, Jones V, Landau Z (2008) A polynomial quantum algorithm for approximating the jones polynomial. Algorithmica 55(3):395–421. https://doi.org/10.1007/s00453-008-9168-0

Arute F, Arya K, Babbush R, Bacon D, Bardin JC, Barends R, Biswas R, Boixo S, Brandao FGSL, Buell DA, Burkett B, Chen Y, Chen Z, Chiaro B, Collins R, Courtney W, Dunsworth A, Farhi E, Foxen B, Fowler A, Gidney C, Giustina M, Graff R, Guerin K, Habegger S, Harrigan MP, Hartmann MJ, Ho A, Hoffmann M, Huang T, Humble TS, Isakov SV, Jeffrey E, Jiang Z, Kafri D, Kechedzhi K, Kelly J, Klimov PV, Knysh S, Korotkov A, Kostritsa F, Landhuis D, Lindmark M, Lucero E, Lyakh D, Mandrà S, McClean JR, McEwen M, Megrant A, Mi X, Michielsen K, Mohseni M, Mutus J, Naaman O, Neeley M, Neill C, Niu MY, Ostby E, Petukhov A, Platt JC, Quintana C, Rieffel EG, Roushan P, Rubin NC, Sank D, Satzinger KJ, Smelyanskiy V, Sung KJ, Trevithick MD, Vainsencher A, Villalonga B, White T, Yao ZJ, Yeh P, Zalcman A, Neven H, Martinis JM (2019) Quantum supremacy using a programmable superconducting processor. Nature 574(7779):505–510. https://doi.org/10.1038/s41586-019-1666-5

Baez JC, Stay M (2009) Physics, topology, Logic and Computation: A Rosetta Stone

Bankova D, Coecke B, Lewis M, Marsden D (2016) Graded entailment for compositional distributional semantics

Bausch J, Subramanian S, Piddock S (2020) A quantum search decoder for natural language processing

Beer K, Bondarenko D, Farrelly T, Osborne TJ, Salzmann R, Scheiermann D, Wolf R (2020) Training deep quantum neural networks. Nature Communications 11(1). https://doi.org/10.1038/s41467-020-14454-2

Benedetti M, Fiorentini M, Lubasch M (2020) Hardware-efficient variational quantum algorithms for time evolution

Benedetti M, Lloyd E, Sack S, Fiorentini M (2019) Parameterized quantum circuits as machine learning models. Quantum Science and Technology 4(4):043001. https://doi.org/10.1088/2058-9565/ab4eb5

Bharti K, Cervera-Lierta A, Kyaw TH, Haug T, Alperin-Lea S, Anand A, Degroote M, Heimonen H, Kottmann JS, Menke T, Mok W-K, Sim S, Kwek L-C, Aspuru-guzik A (2021) Noisy intermediate-scale quantum (NISQ) algorithms

Blackburn P, Bos J (2005) Representation and inference for natural language: a first course in computational semantics center for the study of language and information. Stanford, CA

Bonet-Monroig X, Wang H, Vermetten D, Senjean B, Moussa C, Bäck T, Dunjko V, O'brien TE (2021) Performance comparison of optimization methods on variational quantum algorithms

Bradbury J, Frostig R, Hawkins P, Johnson MJ, Leary C, Maclaurin D, Wanderman-Milne S (2018) JAX: Composable Transformations of Python+NumPy programs. http://github.com/google/jax

Bradley T.-D., Stoudenmire EM, Terilla J (2019) Modeling sequences with quantum states: a look under the hood

Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S, Herbert-Voss A, Krueger G, Henighan T, Child R, Ramesh A, Ziegler DM, 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

Buhrmester V, Münch D, Arens M (2019) Analysis of explainers of black box deep neural networks for computer vision: a survey

Buszkowski W, Moroz K (2007) Pregroup Grammars and Context-free Grammars

Chen JC (2002) Quantum computation and natural language processing

Chen Y, Pan Y, Dong D (2020) Quantum language model with entanglement embedding for question answering

Chia N-H, Gilyén A, Li T, Lin H-H, Tang E, Wang C (2020) Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning. Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing. https://doi.org/10.1145/3357713.3384314

Chomsky N (1957) Syntactic structures. Mouton

Coecke B (2020) The mathematics of text structure

Coecke B, Kissinger A (2017) Picturing quantum processes. a first course in quantum theory and diagrammatic reasoning. Cambridge University Press, Cambridge. https://doi.org/10.1017/9781316219317

Coecke B, Sadrzadeh M, Clark S (2010) Mathematical foundations for a compositional distributional model of meaning

Coecke B, de Felice G, Meichanetzidis K, Toumi A (2020) Foundations for near-term quantum natural language processing

Cowtan A, Dilkes S, Duncan R, Krajenbrink A, Simmons W, Sivarajah S (2019) On the qubit routing problem. In: van Dam W, Mancinska L (eds) 14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019). Leibniz International Proceedings in Informatics (LIPIcs), vol 135. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp 5–1532. https://doi.org/10.4230/LIPIcs.TQC.2019.5. http://drops.dagstuhl.de/opus/volltexte/2019/10397

de Felice G, Meichanetzidis K, Toumi A (2020) Functorial question answering. Electronic Proceedings in Theoretical Computer Science 323:84–94. https://doi.org/10.4204/eptcs.323.6

de Felice G, Toumi A, Coecke B (2020) Discopy: monoidal categories in Python

Dunjko V, Taylor JM, Briegel HJ (2016) Quantum-enhanced machine learning. Physical Review Letters 117(13). https://doi.org/10.1103/physrevlett.117.130501

Efthymiou S, Hidary J, Leichenauer S (2019) Tensornetwork for machine learning

Eisert J (2013) Entanglement and tensor network states

Gallego AJ, Orus R. (2019) Language Design as Information Renormalization

Gao F, Han L (2010) Implementing the nelder-mead simplex algorithm with adaptive parameters. Comput Optim Appl 51(1):259–277. https://doi.org/10.1007/s10589-010-9329-3

Grefenstette E, Sadrzadeh M (2011) Experimental support for a categorical compositional distributional model of meaning. In: The 2014 conference on empirical methods on natural language processing, pp 1394–1404. arXiv:1106.4058

Harrow AW, Hassidim A, Lloyd S (2009) Quantum algorithm for linear systems of equations. Physical Review Letters 103(15). https://doi.org/10.1103/physrevlett.103.150502

Havlíček V, Córcoles AD, Temme K, Harrow AW, Kandala A, Chow JM, Gambetta JM (2019) Supervised learning with quantum-enhanced feature spaces. Nature 567(7747):209–212. https://doi.org/10.1038/s41586-019-0980-2

Jurafsky D, Martin JH (2000) Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, vol 1. Prentice Hall PTR, USA

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

Kartsaklis D, Sadrzadeh M (2013) Prior disambiguation of word tensors for constructing sentence vectors. In: The 2013 conference on empirical methods on natural language processing. ACL, pp 1590–1601

Kerenidis I, Landman J, Luongo A, Prakash A (2019) Q-means: a quantum algorithm for unsupervised machine learning. In: Wallach H, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox E, Garnett R (eds) Advances in neural information processing systems, vol 32. Curran Associates Inc, pp 4134–4144. https://proceedings.neurips.cc/paper/2019/file/16026d60ff9b54410b3435b403afd226-Paper.pdf

Lambek J (1958) The mathematics of sentence structure. American Mathematical Monthly. 154–170

Lambek J (2008) From word to sentence

Lewis M (2020) Towards logical negation for compositional distributional semantics

Li Z, Liu X, Xu N, Du J (2015) Experimental realization of a quantum support vector machine. Physical Review Letters 114(14). https://doi.org/10.1103/physrevlett.114.140504

Lloyd S, Schuld M, Ijaz A, Izaac J, Killoran N (2020) Quantum embeddings for machine learning

Meichanetzidis K, Gogioso S, Felice GD, Chiappori N, Toumi A, Coecke B (2020) Quantum natural language processing on near-term quantum computers

Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space

Mitarai K, Fujii K (2019) Methodology for replacing indirect measurements with direct measurements. Physical Review Research 1(1). https://doi.org/10.1103/physrevresearch.1.013006

Montague R (2008) Universal grammar. Theoria 36(3):373–398. https://doi.org/10.1111/j.1755-2567.1970.tb00434.x

O'Riordan LJ, Doyle M, Baruffa F, Kannan V (2020) A hybrid classical-quantum workflow for natural language processing. Machine Learning: Science and Technology. https://doi.org/10.1088/2632-2153/abbd2e

Olson B, Hashmi I, Molloy K, Shehu A (2012) Basin hopping as a general and versatile optimization framework for the characterization of biological macromolecules. Advances in Artificial Intelligence 2012:1–19. https://doi.org/10.1155/2012/674832

Orús R (2019) Tensor networks for complex quantum systems. Nature Reviews Physics 1(9):538–550. https://doi.org/10.1038/s42254-019-0086-7

Pentus M (1993) Lambek grammars are context free. In: 1993 Proceedings Eighth Annual IEEE Symposium on Logic in Computer Science, pp 429–433. https://doi.org/10.1109/LICS.1993.287565

Pestun V, Vlassopoulos Y (2017) Tensor network language model

Piedeleu R, Kartsaklis D, Coecke B, Sadrzadeh M (2015) Open system categorical quantum semantics in natural language processing

Preller A (2007) Linear processing with pregroups. Studia Logica: An International Journal for Symbolic Logic 87(2/3):171–197

Sadrzadeh M, Clark S, Coecke B (2013) The Frobenius anatomy of word meanings i: subject and object relative pronouns. J Log Comput 23(6):1293–1317. https://doi.org/10.1093/logcom/ext044

Sadrzadeh M, Clark S, Coecke B (2014) The Frobenius anatomy of word meanings ii: possessive relative pronouns. J Log Comput 26(2):785–815. https://doi.org/10.1093/logcom/exu027

Schuld M, Bocharov A, Svore KM, Wiebe N (2020) Circuit-centric quantum classifiers. Physical Review A 101(3). https://doi.org/10.1103/physreva.101.032308

Schuld M, Killoran N (2019) Quantum machine learning in feature hilbert spaces. Physical Review Letters 122(4). https://doi.org/10.1103/physrevlett.122.040504

Searls DB (2002) The language of genes. Nature 420(6912):211–217. https://doi.org/10.1038/nature01255

Selinger P (2010) A survey of graphical languages for monoidal categories. Lecture Notes in Physics, 289–355. https://doi.org/10.1007/978-3-642-12821-9_4

Sivarajah S, Dilkes S, Cowtan A, Simmons W, Edgington A, Duncan R (2020) Tket: a retargetable compiler for nisq devices Quantum Science and Technology. https://doi.org/10.1088/2058-9565/ab8e92

Socher R, Perelygin A, Wu J, Chuang J, Manning CD, Ng A, Potts C (2013) Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 conference on empirical methods in natural language processing. Association for Computational Linguistics, pp 1631–1642. https://www.aclweb.org/anthology/D13-1170

Spall JC (1998) Implementation of the simultaneous perturbation algorithm for stochastic optimization. IEEE Trans Aerosp Electron Syst 34(3):817–823. https://doi.org/10.1109/7.705889

Turing AM (1950) I.—computing machinery and intelligence. Mind LIX(236):433–460. https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf. https://doi.org/10.1093/mind/LIX.236.433

Wiebe N, Bocharov A, Smolensky P, Troyer M, Svore KM (2019) Quantum Language Processing

Wootton JR (2020) Procedural generation using quantum computation. International Conference on the Foundations of Digital Games. https://doi.org/10.1145/3402942.3409600

Yeung R, Kartsaklis D (2021) A CCG-based Version of the DisCoCat Framework

Zeng W, Coecke B (2016) Quantum algorithms for compositional natural language processing. Electronic Proceedings in Theoretical Computer Science 221:67–75. https://doi.org/10.4204/eptcs.221.8

Zeng Z, Shi H, Wu Y, Hong Z (2015) Survey of natural language processing techniques in bioinformatics. Comput Math Methods Med 2015:674296. https://doi.org/10.1155/2015/674296

Zhao Q, Hou C, Liu C, Zhang P, Xu R (2020) A quantum expectation value based language model with application to question answering. Entropy 22(5):533