Cambridge Quantum

Grammar-Aware Question-Answering on Quantum Computers

by Konstantinos Meichanetzidis, Alexi Toumi, Giovanni de Felice, Bob Coecke

Cambridge Quantum
Department of Computer Science
University of Oxford, United Kingdom
KONSTANTINOS MEICHANETZIDIS
konstantinos.meichanetzidis@cs.ox.ac.uk
ALEXIS TOUMI
alexis.toumi@cs.ox.ac.uk
GIOVANNI DE FELICE
giovanni.defelice@cs.ox.ac.uk
BOB COECKE
bob.coecke@cs.ox.ac.uk

Cambridge Quantum
Terrington House
13-15 Hills Road
Cambridge CB2 1NL
United Kingdom

Published by Cambridge Quantum

07 December 2020

Grammar-Aware Question-Answering on Quantum Computers

Konstantinos Meichanetzidis, Alexis Toumi, Giovanni de Felice, Bob Coecke
Cambridge Quantum Computing Ltd. and
Department of Computer Science, University of Oxford

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, 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. Earlier work has already demonstrated potential quantum advantage for NLP in a number of manners: (i) algorithmic speedups for search-related or classification tasks, which are the most dominant tasks within NLP, (ii) exponentially large quantum state spaces allow for accommodating complex linguistic structures, (iii) novel models of meaning employing density matrices naturally model linguistic phenomena such as hyponymy and linguistic ambiguity, among others.

In this work, we perform the first implementation of an NLP task on noisy intermediate-scale quantum (NISQ) hardware. Sentences are instantiated as parameterised quantum circuits. We encode word-meanings in quantum states and we explicitly account for grammatical structure, which even in mainstream NLP is not commonplace, by faithfully hardwiring it as entangling operations. This makes our approach to quantum natural language processing (QNLP) particularly NISQ-friendly. Our novel QNLP model shows concrete promise for scalability as the quality of the quantum hardware improves in the near future.

NLP is a rapidly evolving area of AI of both theoretical importance and practical interest [1, 2]. State of the art language models, such as GPT-3 with 175 billion parameters [3], show impressive results on general NLP tasks and one dares to claim that humanity is entering Turing-test territory [4]. 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 [5, 6]. 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 [7].

One way to bring transparency to said black boxes, is to incorporate linguistic structure [8–10] into distributional language models. A prominent approach attempting this merge is the Distributional Compositional Categorical model of natural language meaning (DisCoCat) [11–13], which pioneered the paradigm of combining grammatical (or syntactic) structure and statistical methods for encoding and computing meaning (or semantics). 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 [14, 15], conjuction, disjuction, and negation [16]. From a modern lens, DisCoCat 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 [17–21].

Quantum computing (QC) is a field which, in parallel with NLP is growing at an extremely fast pace. The importance of QC is now well-established, especially after the demonstration of quantum supremacy [22], and reaches the whole range of human interests, from foundations of physics and computer science, to applications in engineering, finance, chemistry, and optimisation problems.

In the last half-decade, the natural conceptual fusion of QC with AI, and especially 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 [23], or quantising classical machine learning algorithms based on neural networks [24], support vector machines, clustering [25], or artificial agents who learn from interacting with their environment [26], and even quantum-inspired and dequantised classical algorithms which nevertheless retain a complexity theoretic advantage [27]. Small-scale classification experiments have also been implemented with quantum technology [28, 29].

From this collection of ingredients there organically emerges the interdisciplinary field of Quantum Natural Language Processing (QNLP), a research area still in its infancy [30–34] QNLP combines NLP and QC and seeks algorithms and novel quantum language models. Building on the recently established methodology of QML, one aims to import 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, by Zeng and Coecke [30], introduced an approach to QNLP where NLP tasks modelled in the DisCoCat framework are instantiated as quantum computations, and, remarkably, a quadratic speedup was obtained for the task of sentence similarity. The mapping's simplicity is attributed to the mathematical similarity of the structures underlying DisCoCat and quantum theory, the latter as formulated by categorical quantum mechanics [35]. This similarity becomes apparent when both are expressed in the graphical language of string diagrams of monoidal categories or process theories [36], which are equivalent to tensor networks. This is what makes DisCoCat 'quantum-native'.

In this work, we bring quantum DisCoCat to the current age of noisy intermediate-scale quantum (NISQ) devices [37] by adopting the paradigm of PQCs [38, 39]. We argue that DisCoCat can justifiably be termed as "NISQ-friendly", as it allows for the execution of proof-of-concept experiments involving a non-trivial corpus, which moreover involves complex grammatical structures. We present results on the first-ever QNLP experiment on NISQ devices.

The model: DisCoCat is based on the algebraic model of pregroup grammars (Appendix A) developed by Lambek [40]. A sentence is a finite product of words $\sigma = \prod_i w_i$. A parser tags a word $w \in \sigma$ with its part of speech.

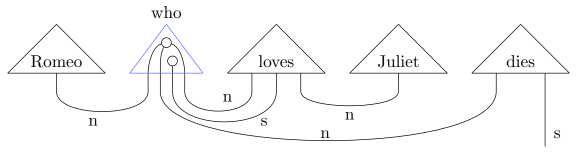

FIG. 1. Diagram for "Romeo who loves Juliet dies". The grammatical reduction is generated by the nested pattern of non-crossing cups, which connect words through wires of types n or s. Grammaticality is verified by only one s-wire left open. The diagram represents the meaning of the whole sentence from 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.

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 [41]. The type of a sentence is the product of the types of its words and it is deemed grammatical iff it type-reduces to the special type $s^0 \in B$, i.e. the sentence-type, $t_\sigma = \prod_w t_w \to s^0$. Reductions are performed by iteratively applying pairwise annihilations of a basic types with adjoint orders of the form $b^i b^{i+1}$. As an example consider the reduction: $t_{\text{Romeo who loves Juliet dies}} = t_{\text{Romeo}}\, t_{\text{who}}\, t_{\text{loves}}\, t_{\text{Juliet}}\, t_{\text{dies}} = (n^0)(n^1 n^0 s^{-1} n^0)(n^1 s^0 n^{-1})(n^0)(n^1 s^0) \to \cancel{n^0 n^1} n^0 s^{-1} \cancel{n^0 n^1} s^0 \cancel{n^{-1} n^0} n^1 s^0 \to n^0 \cancel{s^{-1} s^0} n^1 s^0 \to \cancel{n^0 n^1} s^0 \to s^0$.

Crucial for our work is acknowledging that 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 [42]. Process networks such as those that manifest in DisCoCat can be represented graphically as string diagrams [43]. String diagrams are not just convenient graphical notation, but they constitute a formal graphical type-theoretic language for reasoning about complex process networks (Appendix B), and are key to our QNLP methods. 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 pro-

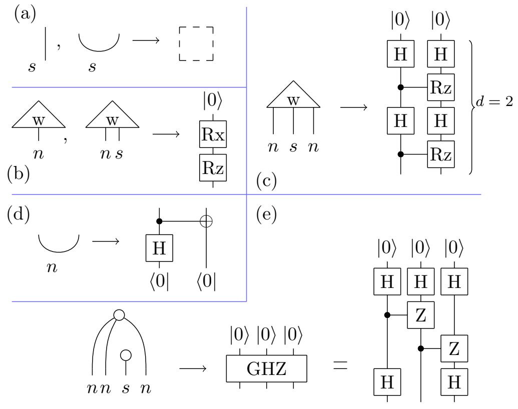

FIG. 2. Example instance of mapping from sentence diagrams to PQCs where $q_n = 1$ and $q_s = 0$. (a) The dashed square is the empty diagram. In this example, (b) unary word-states are prepared by parameterised R_x rotations followed by R_z rotations and (c) k-ary word-states are prepared by parameterised 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.

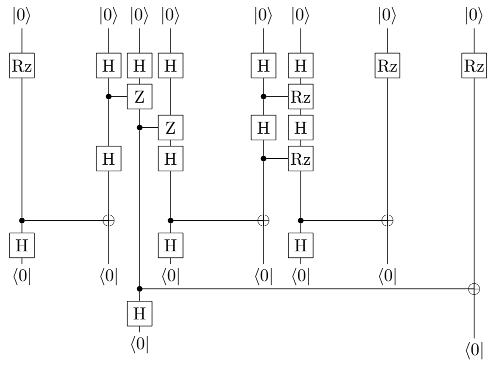

FIG. 3. The PQC to which "Romeo who loves Juliet dies" of Fig.1 is mapped, with the choices of hyperparameters of Fig.2. As $q_s = 0$, the circuit represents a scalar.

cess and so it can be represented as a string diagram. Words are represented as states and pairwise type-reductions are represented by a pattern of nested cups-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. 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.*

As described in Ref.[37], the diagram 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 $|w(\theta_w)\rangle = C_w(\theta_w)|0\rangle^{\otimes q_w}$. 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, $q_w = \sum_{b \in t_w} q_b$, 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 [28, 44, 45] 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 as its word-circuit the Euler parametrisation $\mathrm{R}_z(\theta_w^3)R_x(\theta_w^2)$.For a word w of arity $k \geq 2$, we use a depth-d IQP-style parameterisation [28] consisting of d layers where each layer consists of a layer of Hadamard gates followed by a lower of controlled-Z rotations $\mathrm{CR}_z(\theta_w^i)$, 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 [28]. 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 [14, 15].

In Fig.2 we show an example of choices of word-circuits for specific numbers of qubits assigned to each basic pregroup type (Appendix 8). In Fig.3 we show the corresponding circuit to "Romeo who loves Juliet dies".

Here, a motivating remark is in order. In classical implementations of DisCoCat, sentence diagrams represent tensor contractions. Meanings of words are encoded in terms of cooccurrence frequencies or other vector-space word-embeddings such as those produced by neural networks [46]. Tensor contractions are exponentially expensive in the dimension of the vector spaces carried by the wires, which for NLP applications can become prohibitively large. In the quantum case, however, the tensor product structure defined by a collection of qubits provides an exponentially large Hilbert space, leading to *exponential space-gain*. Consequently, we adopt the paradigm of QML in terms of PQCs to carry out near-term QNLP tasks.

Question Answering: 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 $l_\sigma \in \{0,1\}$ interpreted as its truth value. 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.

The sentences are generated randomly using a context-free grammar (CFG). Each sentence is represented as a syntax tree, which also can be cast in string-diagram form. Each CFG-generated sentence diagram can then be canonically transformed into a DisCoCat diagram (Appendix C). 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, rendering the QA task non-trivial.

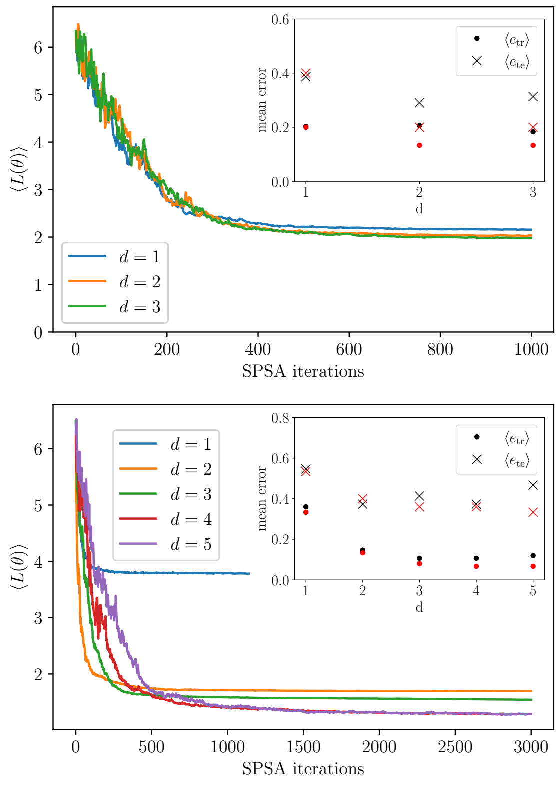

FIG. 4. Convergence of mean cost function $\langle L(\theta)\rangle$ vs number of `SPSA` iterations for corpus K_{30}. A lower minimum is reached for larger d. (Top) $q_n = 1$ and $|\theta| = 8+2d$. Results are averaged over 20 realisations. (Bottom) $q_n = 2$ and $|\theta| = 10d$. Results are averaged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

We define the *predicted label* as

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

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

Now 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` [47], which has shown adequate performance in noisy settings (Appendix

F). The *cost function* we define is

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

Minimising the cost function returns the optimal parameters $\theta^* = \text{argmin} L(\theta)$ from which the model predicts the labels $l_\sigma^{\rm pr}(\theta^*)$. Essentially, this constitutes *learning a functor* from the grammar category to the categeory 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:

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

This supervised learning task of binary classification for sentences is a special case of *question answering* (QA) [48–50]. Questions are posed as statements and the truth labels are the 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 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, that every word appears in at least one sentence, which introduces correlations between the sentences and makes such a learning task possible.

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).

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,

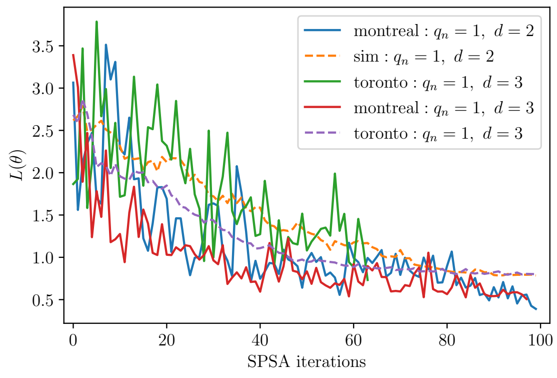

FIG. 5. Convergence of the cost $L(\theta)$ evaluated on quantum computers vs `SPSA` iterations for corpus K_{16}. For $q_n = 1$, $d = 2$, for which $|\theta| = 10$, on `ibmq_montreal` (blue) we obtain $e_{\rm tr} = 0.125$ and $e_{\rm te} = 0.5$. For $q_n = 1$, $d = 3$, where $|\theta| = 13$, on `ibmq_toronto` (green) we get $e_{\rm tr} = 0.125$ and a lower testing error $e_{\rm te} = 0.375$. On `ibmq_montreal` (red) we get both lower training and testing errors, $e_{\rm tr} = 0$, $e_{\rm te} = 0.375$ than for $d = 2$. In all cases, the CNOT-depth of any sentence-circuit after t|ket⟩-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).

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 t|ket⟩ [51] by Cambridge Quantum Computing (CQC), 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.

Discussion and Outlook: We have performed the first-ever quantum natural language processing experiment by means of question answering on quantum hardware. We used a compositional model of meaning constructed by a structure-preserving mapping from grammatical sentences to PQCs.

Here 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, evaluating a scalar encoded in an amplitude on a quantum computer requires either postselection or coherent control over arbitrary circuits so that a Hadamard test [52] can be performed (Appendix H). Notably, in special cases of interest to QML, the Hadamard test can be adapted to NISQ technologies [53, 54]. 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 implementation of more complex QNLP tasks such as sentence similarity and work with real-world data using a pregroup parser. In that context, we plan to explore regularisation techniques for the QML aspect of our work, which is an increasingly relevant topic that in general deserves more attention [39]. 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 entailement and ambiguity [55, 56].

Finally, looking beyond the DisCoCat model, it is well-motivated to adopt the recently introduced 'augmented' DisCoCirc model [57] of meaning and its mapping to PQCs [58], which allows for QNLP experiments on text-scale real-world data in a fully-compositional framework. Motivated by interpretability in AI, word-meanings in DisCoCirc are built bottom-up as parameterised states defined on specific tensor factors. Nouns are treated as '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. This interaction structure, viewed as a process network, can be mapped to quantum circuits, with entities as density matrices carried by wires and their modifiers as quantum channels.

Acknowledgments KM thanks Vojtech Havlicek and Chris Self for discussions on QML, Robin Lorenz and Marcello Benedetti for comments on the manuscript, and the t|ket⟩ team at CQC for technical advice on quantum compilation. 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.

[1] Daniel Jurafsky and James H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 1st ed. (Prentice Hall PTR, USA, 2000).

[2] Patrick Blackburn and Johan Bos, *Representation and Inference for Natural Language: A First Course in Computational Semantics* (Center for the Study of Language and Information, Stanford, CA, 2005).

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei, "Language models are few-shot learners," (2020), arXiv:2005.14165 [cs.CL].

[4] A. M. TURING, "I.—COMPUTING MACHINERY AND INTELLIGENCE," Mind **LIX**, 433–460 (1950), https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf.

[5] David B. Searls, "The language of genes," Nature **420**, 211–217 (2002).

[6] Zhiqiang Zeng, Hua Shi, Yun Wu, and Zhiling Hong, "Survey of natural language processing techniques in bioinformatics," Computational and Mathematical Methods in Medicine **2015**, 674296 (2015).

[7] Vanessa Buhrmester, David Münch, and

Michael Arens, "Analysis of explainers of black box deep neural networks for computer vision: A survey," (2019), arXiv:1911.12116 [cs.AI].

[8] Joachim Lambek, "The mathematics of sentence structure," AMERICAN MATHEMATICAL MONTHLY , 154–170 (1958).

[9] RICHARD MONTAGUE, "Universal grammar," Theoria **36**, 373–398 (2008).

[10] Noam Chomsky, *Syntactic Structures* (Mouton, 1957).

[11] Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark, "Mathematical foundations for a compositional distributional model of meaning," (2010), arXiv:1003.4394 [cs.CL].

[12] E. Grefenstette and M. Sadrzadeh, "Experimental support for a categorical compositional distributional model of meaning," in *The 2014 Conference on Empirical Methods on Natural Language Processing.* (2011) pp. 1394–1404, arXiv:1106.4058.

[13] D. Kartsaklis and M. Sadrzadeh, "Prior disambiguation of word tensors for constructing sentence vectors." in *The 2013 Conference on Empirical Methods on Natural Language Processing.* (ACL, 2013) pp. 1590–1601.

[14] M. Sadrzadeh, S. Clark, and B. Coecke, "The frobenius anatomy of word meanings i: subject and object relative pronouns," Journal of Logic and Computation **23**, 1293–1317 (2013).

[15] Mehrnoosh Sadrzadeh, Stephen Clark, and Bob Coecke, "The frobenius anatomy of word meanings ii: possessive relative pronouns," Journal of Logic and Computation **26**, 785–815 (2014).

[16] Martha Lewis, "Towards logical negation for compositional distributional semantics," (2020), arXiv:2005.04929 [cs.CL].

[17] Vasily Pestun and Yiannis Vlassopoulos, "Tensor network language model," (2017), arXiv:1710.10248 [cs.CL].

[18] Angel J. Gallego and Roman Orus, "Language design as information renormalization," (2019), arXiv:1708.01525 [cs.CL].

[19] Tai-Danae Bradley, E. Miles Stoudenmire, and John Terilla, "Modeling sequences with quantum states: A look under the hood," (2019), arXiv:1910.07425 [quant-ph].

[20] Stavros Efthymiou, Jack Hidary, and Stefan Leichenauer, "Tensornetwork for machine learning," (2019), arXiv:1906.06329 [cs.LG].

[21] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts, "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, Seattle, Washington, USA, 2013) pp. 1631–1642.

[22] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis, "Quantum supremacy using a programmable superconducting processor," Nature **574**, 505–510 (2019).

[23] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd, "Quantum algorithm for linear systems of equations," Physical Review Letters **103** (2009), 10.1103/physrevlett.103.150502.

[24] Kerstin Beer, Dmytro Bondarenko, Terry Farrelly, Tobias J. Osborne, Robert Salzmann, Daniel Scheiermann, and Ramona Wolf, "Training deep quantum neural networks," Nature Communications **11** (2020), 10.1038/s41467-020-14454-2.

[25] Iordanis Kerenidis, Jonas Landman, Alessandro Luongo, and Anupam Prakash, "q-means: A quantum algorithm for unsupervised machine learning," in *Advances in Neural Information Processing Systems*, Vol. 32, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019) pp. 4134–4144.

[26] Vedran Dunjko, Jacob M. Taylor, and Hans J. Briegel, "Quantum-enhanced machine learning," Physical Review Letters **117** (2016), 10.1103/physrevlett.117.130501.

[27] Nai-Hui Chia, András Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang, "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 (2020), 10.1145/3357713.3384314.

[28] Vojtěch Havlíček, Antonio D. Córcoles, Kristan

Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," Nature **567**, 209–212 (2019).

[29] Zhaokai Li, Xiaomei Liu, Nanyang Xu, and Jiangfeng Du, "Experimental realization of a quantum support vector machine," Physical Review Letters **114** (2015), 10.1103/physrevlett.114.140504.

[30] William Zeng and Bob Coecke, "Quantum algorithms for compositional natural language processing," Electronic Proceedings in Theoretical Computer Science **221**, 67–75 (2016).

[31] Lee James O'Riordan, Myles Doyle, Fabio Baruffa, and Venkatesh Kannan, "A hybrid classical-quantum workflow for natural language processing," Machine Learning: Science and Technology (2020), 10.1088/2632-2153/abbd2e.

[32] Nathan Wiebe, Alex Bocharov, Paul Smolensky, Matthias Troyer, and Krysta M Svore, "Quantum language processing," (2019), arXiv:1902.05162 [quant-ph].

[33] Johannes Bausch, Sathyawageeswar Subramanian, and Stephen Piddock, "A quantum search decoder for natural language processing," (2020), arXiv:1909.05023 [quant-ph].

[34] Joseph CH Chen, "Quantum computation and natural language processing," (2002).

[35] S. Abramsky and B. Coecke, "A categorical semantics of quantum protocols," in *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004.* (2004) pp. 415–425.

[36] B. Coecke and A. Kissinger, *Picturing Quantum Processes. A First Course in Quantum Theory and Diagrammatic Reasoning* (Cambridge University Press, 2017).

[37] Konstantinos Meichanetzidis, Stefano Gogioso, Giovanni De Felice, Nicolò Chiappori, Alexis Toumi, and Bob Coecke, "Quantum natural language processing on near-term quantum computers," (2020), arXiv:2005.04147 [cs.CL].

[38] Maria Schuld, Alex Bocharov, Krysta M. Svore, and Nathan Wiebe, "Circuit-centric quantum classifiers," Physical Review A **101** (2020), 10.1103/physreva.101.032308.

[39] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini, "Parameterized quantum circuits as machine learning models," Quantum Science and Technology **4**, 043001 (2019).

[40] Joachim Lambek, "From word to sentence," .

[41] Anne Preller, "Linear processing with pregroups," Studia Logica: An International Journal for Symbolic Logic **87**, 171–197 (2007).

[42] John C. Baez and Mike Stay, "Physics, topology, logic and computation: A rosetta stone," (2009), arXiv:0903.0340 [quant-ph].

[43] P. Selinger, "A survey of graphical languages for monoidal categories," Lecture Notes in Physics , 289–355 (2010).

[44] Maria Schuld and Nathan Killoran, "Quantum machine learning in feature hilbert spaces," Physical Review Letters **122** (2019), 10.1103/physrevlett.122.040504.

[45] Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac, and Nathan Killoran, "Quantum embeddings for machine learning," (2020), arXiv:2001.03622 [quant-ph].

[46] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, "Efficient estimation of word representations in vector space," (2013), arXiv:1301.3781 [cs.CL].

[47] J. C. Spall, "Implementation of the simultaneous perturbation algorithm for stochastic optimization," IEEE Transactions on Aerospace and Electronic Systems **34**, 817–823 (1998).

[48] Giovanni de Felice, Konstantinos Meichanetzidis, and Alexis Toumi, "Functorial question answering," Electronic Proceedings in Theoretical Computer Science **323**, 84–94 (2020).

[49] Yiwei Chen, Yu Pan, and Daoyi Dong, "Quantum language model with entanglement embedding for question answering," (2020), arXiv:2008.09943 [cs.CL].

[50] Qin Zhao, Chenguang Hou, Changjian Liu, Peng Zhang, and Ruifeng Xu, "A quantum expectation value based language model with application to question answering," Entropy **22**, 533 (2020).

[51] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan, "t—ket⟩: A retargetable compiler for nisq devices," Quantum Science and Technology (2020), 10.1088/2058-9565/ab8e92.

[52] Dorit Aharonov, Vaughan Jones, and Zeph Landau, "A polynomial quantum algorithm for approximating the jones polynomial," Algorithmica **55**, 395–421 (2008).

[53] Kosuke Mitarai and Keisuke Fujii, "Methodology for replacing indirect measurements with direct measurements," Physical Review Research **1** (2019), 10.1103/physrevresearch.1.013006.

[54] Marcello Benedetti, Mattia Fiorentini, and Michael Lubasch, "Hardware-efficient variational quantum algorithms for time evolution," (2020), arXiv:2009.12361 [quant-ph].

[55] Robin Piedeleu, Dimitri Kartsaklis, Bob Coecke, and Mehrnoosh Sadrzadeh, "Open system categorical quantum semantics in natural language processing," (2015), arXiv:1502.00831 [cs.CL].

[56] Desislava Bankova, Bob Coecke, Martha Lewis, and Daniel Marsden, "Graded entailment for compositional distributional semantics," (2016), arXiv:1601.04908 [cs.CL].

[57] Bob Coecke, "The mathematics of text structure," (2020), arXiv:1904.03478 [cs.CL].

[58] Bob Coecke, Giovanni De Felice, Konstantinos Meichanetzidis, and Alexis Toumi, "Foundations for near-term quantum natural language processing (unpublished)," (2020).

[59] Wojciech Buszkowski and Katarzyna Moroz, “Pregroup grammars and context-free grammars,”.

[60] M. Pentus, “Lambek grammars are context free,” in *[1993] Proceedings Eighth Annual IEEE Symposium on Logic in Computer Science* (1993) pp. 429–433.

[61] https://github.com/andim/noisyopt.

[62] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne, “JAX: composable transformations of Python+NumPy programs,” (2018).

[63] Brian Olson, Irina Hashmi, Kevin Molloy, and Amarda Shehu, “Basin hopping as a general and versatile optimization framework for the characterization of biological macromolecules,” Advances in Artificial Intelligence **2012**, 1–19 (2012).

[64] Fuchang Gao and Lixing Han, “Implementing the nelder-mead simplex algorithm with adaptive parameters,” Computational Optimization and Applications **51**, 259–277 (2010).

[65] https://pypi.org/project/scipy.

[66] Alexander Cowtan, Silas Dilkes, Ross Duncan, Alexandre Krajenbrink, Will Simmons, and Seyon Sivarajah, “On the Qubit Routing Problem,” in *14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019)*, Leibniz International Proceedings in Informatics (LIPIcs), Vol. 135, edited by Wim van Dam and Laura Mancinska (Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2019) pp. 5:1–5:32.

[67] https://github.com/CQCL/pytket.

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, t|ket⟩.

Appendix A: Pregroup Grammar

Pregroup grammars where introduced by Lambek as an algebraic model for grammar [40].

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 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 $T \to \prod_i T_i$, 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

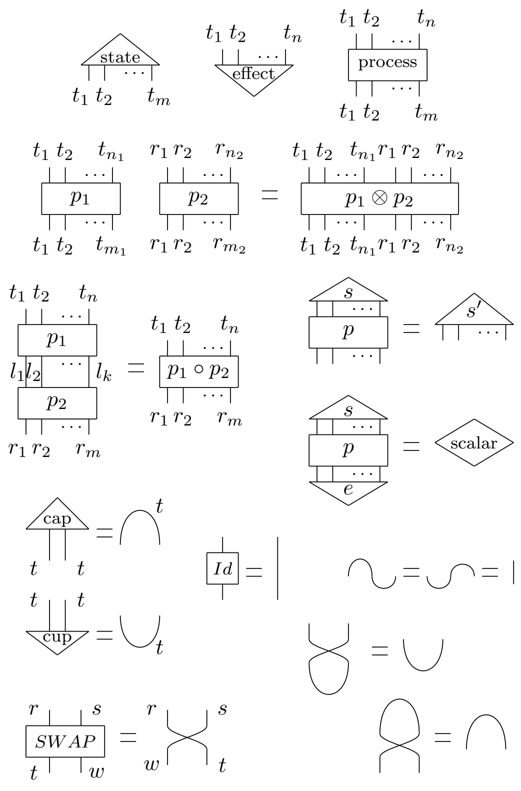

FIG. 6. Diagrams are read from top to bottom. States have only outputs, effects have only inputs, processes (boxes) have both input and output wires. All wires carry types. Placing boxes side by side is allowed by the monoidal structure and signifies parallel processes. Sequential process composition 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.

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, $V = V_N \cup V_{TV} \cup V_{IV} \cup V_{RPRON}$.

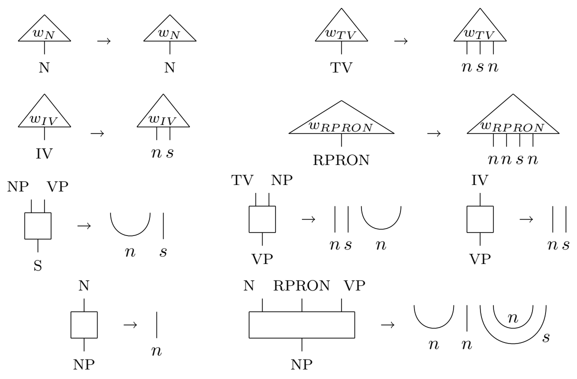

FIG. 7. CFG generation rules used to produce the corpora K_{30}, K_6, K_{16} used in this work, represented as string-diagram generators, where $w_N \in V_N$, $w_{TV} \in V_{TV}$, $w_{IV} \in V_{IV}$, $w_{RPRON} \in V_{RPRON}$. They are mapped to pregroup reductions by mapping CFG symbols to pregroup 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.

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 $b^k b^{k+1} \to \epsilon$. Pregroup grammars are weakly equivalent to context-free grammars, in the sense that they generate the same language [59, 60].

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 = {'Dude', 'Walter'}, V_{TV} = {'loves', 'annoys'}, V_{IV} = {'abides','bowls'}, V_{RPRON} = {'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 = {'Romeo', 'Juliet'}, V_{TV} = {'loves'}, V_{IV} = {'dies'}, V_{RPRON} = {'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 = {'Romeo', 'Juliet'}, V_{TV} = {'loves', 'kills'}, V_{IV} = {'dies'}, V_{RPRON} = {'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 $CR_z(\theta)$ gate, $\big(\otimes_{i=1}^{k} H\big) \circ \big(\otimes_{i=1}^{k-1} CR_z(\theta_i)_{i,i+1}\big)$. 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

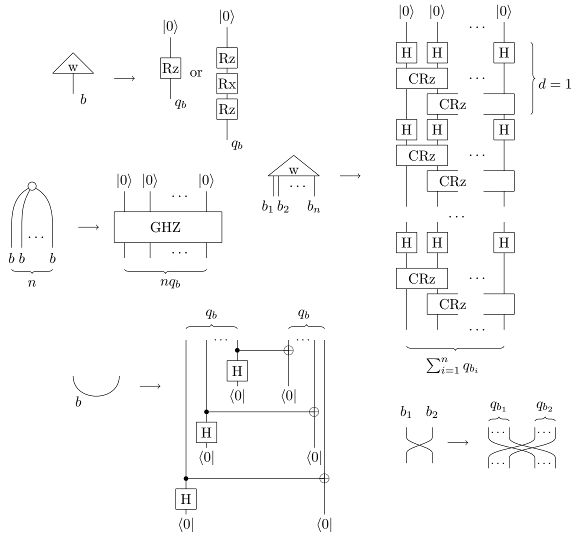

FIG. 8. Mapping from sentence diagrams to parameterised quantum circuits. Here we show how the generators of sentence diagrams are mapped to generators of circuits, for the hyperparameters we consider in this work.

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` [61], 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` [62]. 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

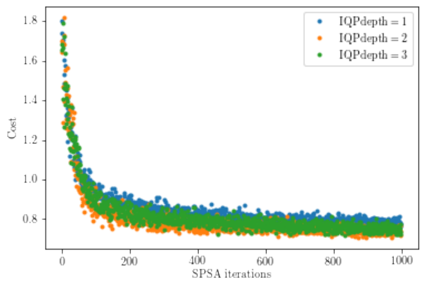

FIG. 9. Minimisation of binary cross entropy cost function L^{BCE} with `SPSA` for the question answering task for corpus K_{30}.

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

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` [63] in combination with `Nelder-Mead` [64] from the Python package `SciPy` [65]. `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 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).

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.

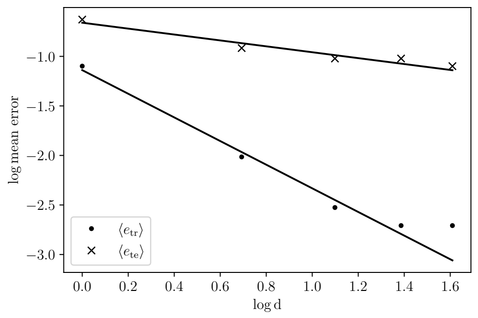

FIG. 10. Algebraic decay of mean training and testing error for the data displayed in Fig.4 (bottom) obtained by `basinhopping`. Increasing the depth of the word-circuits results in algebraic decay of the mean training and testing errors. The slopes are $\log e_{\mathrm{tr}} \sim \log -1.2d$ and $\log e_{\mathrm{te}} \sim -0.3 \log d$. 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.

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` [51]. It is a Python module for interfacing with CQC's t|ket⟩, 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 [66]. Specifically, the compilation pass that we use is `default_compilation_pass(2)`. The integer option is set to 2 for maximum optimisation under compilation [67].

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 t|ket⟩ 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}.

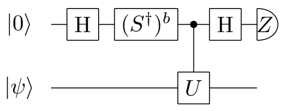

FIG. 11. Circuit for the Hadamard test. Measuring the control qubit in the computational basis allows one to estimate $\langle Z \rangle = \mathrm{Re}(\langle\psi|U|\psi\rangle)$ if $b = 0$, and $\langle Z \rangle = \mathrm{Im}(\langle\psi|U|\psi\rangle)$ if $b = 1$. The state ψ can be a multiqubit state,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

Appendix H: Hadamard Test

In our binary classification NLP task, the predicted label is the norm squared of zero-to-zero transition amplitude $\langle 0 \ldots 0|U|0 \ldots 0\rangle$, where the unitary U includes the word-circuits and the circuits that implement the Bell effects as dictated by the grammatical structure. Estimating these amplitudes can be done by postselecting on $\langle 0 \ldots 0|$. However, postselection costs exponential time in the number of postselected qubits; in our case needs to discard all bitstring sampled from the quantum computer that have Hamming weight other than zero. This is the procedure we

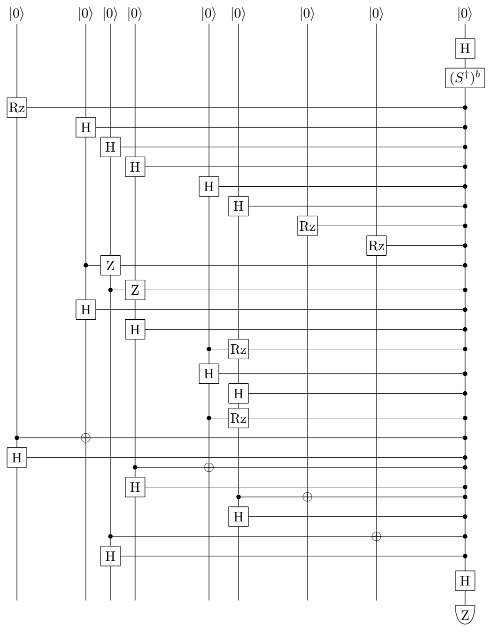

FIG. 12. Use of Hadamard test to estimate the amplitude represented by the postselected circuit of Fig.3.

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 Hadamard test [52]. See Fig.11 for the circuit allowing for the estimation of the real and imaginary part of an amplitude. In Fig.12 we show how the Hadamard test can be used to estimate the amplitude represented by the postselected quantum circuit of Fig.3.