Journal of Artificial Intelligence Research 76 (2023) 1305-1342 Submitted 10/2022; published 04/2023

QNLP in Practice: Running Compositional Models of Meaning on a Quantum Computer

Robin Lorenz robin.lorenz@quantinuum.com
Anna Pearson anna.pearson@quantinuum.com
Konstantinos Meichanetzidis k.mei@quantinuum.com
Dimitri Kartsaklis dimitri.kartsaklis@quantinuum.com
Bob Coecke bob.coecke@quantinuum.com
Quantinuum LLC
17 Beaumont Street, Oxford, OX1 2NA, UK

Abstract

Quantum Natural Language Processing (QNLP) deals with the design and implementation of NLP models intended to be run on quantum hardware. In this paper, we present results on the first NLP experiments conducted on Noisy Intermediate-Scale Quantum (NISQ) computers for datasets of size greater than 100 sentences. Exploiting the formal similarity of the compositional model of meaning by Coecke, Sadrzadeh, and Clark (2010) with quantum theory, we create representations for sentences that have a natural mapping to quantum circuits. We use these representations to implement and successfully train NLP models that solve simple sentence classification tasks on quantum hardware. We conduct quantum simulations that compare the syntax-sensitive model of Coecke et al. with two baselines that use less or no syntax; specifically, we implement the quantum analogues of a "bag-of-words" model, where syntax is not taken into account at all, and of a word-sequence model, where only word order is respected. We demonstrate that all models converge smoothly both in simulations and when run on quantum hardware, and that the results are the expected ones based on the nature of the tasks and the datasets used. Another important goal of this paper is to describe in a way accessible to AI and NLP researchers the main principles, process and challenges of experiments on quantum hardware. Our aim in doing this is to take the first small steps in this unexplored research territory and pave the way for practical Quantum Natural Language Processing.

1. Introduction

With the potential to provide computational speedups over the current standard, quantum computing has rapidly evolved to become one of the most popular cutting-edge areas in computer science, with promising results and applications spanning a wide range of topics such as cryptography (Pirandola et al., 2020), chemistry (Cao et al., 2019), and biomedicine (Cao et al., 2018). An obvious question is whether this new paradigm of computation can also be used for NLP. Such applicability may be to the end of leveraging the computational speedups for language-related problems, as well as for investigating how quantum systems, their mathematical description and the way information is encoded "quantumly" may lead to conceptual and practical advances in representing and processing language meaning beyond computational speedups.

Inspired by these prospects, *Quantum Natural Language Processing* (QNLP), a field of research still in its infancy, aims at the development of NLP models explicitly designed to

be executed on quantum hardware. There exists some impressive theoretical work in this area, but the proposed experiments are classically[1] simulated. A notable exception to this is a work by (Meichanetzidis et al., 2023), where a proof of concept experiment with a dataset of 16 sentences was performed on quantum hardware for the first time.

In this paper we take a significant step further and present two complete experiments consisting of linguistically-motivated NLP tasks with datasets of the order of 100-150 sentences running on quantum hardware. The goal of these experiments is not to demonstrate some form of "quantum advantage" over classical implementations of NLP tasks; for any practical application, this is not yet possible due to the limited capabilities of the currently available quantum computers. In this work, we are mostly interested in exploring the process of running NLP models on quantum hardware, and in providing a detailed account to the AI and NLP communities of what QNLP entails in practice. We show how the traditional modelling and coding paradigm can shift to a quantum-friendly form, and we take a closer look at the challenges and limitations imposed by the current *Noisy Intermediate-Scale Quantum* (NISQ) computers.

From an NLP perspective, both of the tasks that this work considers involve some form of sentence classification: for each sentence in the dataset, we apply compositional models with various degrees of syntax sensitivity to compute a state vector, which is then converted to a binary label. The models are trained on a standard binary cross entropy objective, using an optimisation technique known as SPSA – *Simultaneous Perturbation Stochastic Approximation* (Spall, 1998). We implement quantum versions of the following models:

- a bag-of-words model, where a sentence is represented as an unordered set of words, with no syntactic information present whatsoever.
- a word-sequence model, in which the sentence is processed in a linear manner from left to right, as in a standard recurrent neural network. Since the model respects the order of the words, a limited degree of syntax is captured in this case.
- a fully syntax-based model where composition takes place following a grammatical derivation given by a syntax tree.

For the syntax-based case, which is admittedly the most complicated and interesting one from both a theoretical and an engineering point of view, we use the compositional model of Coecke et al. (2010) – often dubbed DisCoCat (DIStributional COmpositional CATegorical). The choice of DisCoCat is motivated by the fact that the derivations it produces essentially form a tensor network, which means they are already very close to how quantum computers process data. Furthermore, the model comes with a rigorous treatment of the interplay between syntax and semantics and with a convenient diagrammatic language. In Section 5 we will see how the produced diagrams get a natural translation to quantum circuits – the basic units of computation on a quantum computer[2] – and how sentences of different grammatical structure are mapped to different quantum circuits. We further discuss the role of noise on a NISQ device, and how this affects our design choices

1. In this paper the usage of the term 'classical' is that of a qualifier in the sense of classical mechanics or classical probability theory, i.e. in contrast to quantum mechanics.
2. In the circuit-based, as opposed to the measurement-based, model of quantum computation.

for the circuits. The experiments are performed on an IBM NISQ computer provided by the IBM Quantum platform[3].

For our experiments we use two different datasets. The first one (130 sentences) was generated automatically by a simple context-free grammar, with half of the sentences related to food and half related to IT (a binary classification task). The other one (105 noun phrases) was extracted from the RelPron dataset (Rimell et al., 2016), and the goal of the model is to predict whether a noun phrase contains a subject-based or an object-based relative clause (again a binary classification task). As we will see later, although by any meaningful NLP standard the selected datasets and tasks are small-scale and rather trivial in nature, they already pose significant challenges for the current NISQ computers.

Despite the limitations, we demonstrate that all models converge smoothly, and that they produce good results (given the size of the datasets) in both simulated and quantum hardware runs. As a sanity check, a comparison of the performances of the various models reveals meaningful correlations between the degree of syntax sensitivity of the models and the tasks at hand: for example, the amount of syntax required to solve the meaning classification task is minimal, since a simple examination of each word in isolation is usually sufficient to produce the correct classification label for the sentence. This is however not true for the relative pronoun task, where syntactical structure starts to become important. The results we report follow and confirm these intuitions, showing that the bag-of-words model and the word-sequence model perform better on the former task, while DisCoCat presents the best performance on the latter, as expected.

In summary, the contributions of this paper are the following:

- we propose quantum versions for a number of compositional NLP models;
- we outline in some depth the process, the technicalities and the challenges of training and running an NLP model on a quantum computer;
- we present the first set of meaningful NLP experiments on quantum hardware, providing a strong proof of concept that quantum NLP is within our reach.

The structure of the paper is the following: Section 2 discusses the most important related work on experimental quantum computing, particularly with regard to Quantum Machine Learning (QML) and QNLP; Section 3 first briefly discusses NLP models for sentences in general and then describes in detail the DisCoCat model, as well as quantum-friendly models for sentence representation based on the word-sequence and bag-of-words models; Section 4 provides an introduction to quantum computing; Section 5 gives a high-level overview for a general QNLP pipeline; Section 6 explains the tasks; Section 7 provides all the necessary details for the experiments, and presents our results, and finally, Section 8 summarises our findings and points to future work.

2. Related Work

On the matter of quantum computation in the NISQ era generally speaking, there is a plethora of hybrid classical-quantum algorithms with NISQ technology in mind (for a review

3. `https://quantum-computing.ibm.com`

see Bharti et al., 2022; Preskill, 2018). These typically take the form of quantum machine learning (QML) protocols, the majority of which are based on *variational quantum circuit* methods, where the parameters of a quantum circuit are trained through machine learning methods (see, e.g., Benedetti et al., 2019; Jaderberg et al., 2022; Harrow & Napp, 2021; Huang et al., 2021; Ferguson et al., 2021). However, useful quantum algorithms with theoretically proven speedups rely on fault-tolerant quantum computers, which are currently not available. (See Section 4 for more details on this aspect and basic concepts of quantum computation.)

Turning to works on quantum algorithms that specifically address language related problems (without employing the kind of model that this present work does) Bausch et al. (2021) utilise Grover search to achieve superpolynomial speedups for the parsing problem, Wiebe et al. (2019) use quantum annealing to solve problems for general context-free languages and Ramesh and Vinay (2003) provide quantum speedups for string-matching which is relevant for language recognition. Furthermore, in (Gallego & Orús, 2022) parse trees are interpreted as an information-coarse-graining of tensor networks and it is also proposed that they can be instantiated as quantum circuits. To the best of our knowledge, any known quantum algorithm on language related problems with a proven speedup, including those listed above, is not implementable on today's NISQ technology.

Works that are not concerned with an implementation on quantum hardware, but study classical models for natural language that may be seen as inspired by certain aspects of the mathematical formalism of quantum theory are abundant, but peacemeal. See, for instance, (Basile & Tamburini, 2017) and (Chen et al., 2021). While not directly related to NLP, it is worth noting that there also is a lot of interesting work on quantum neural networks, see, for example (Gupta & Zia, 2001; Beer et al., 2020).

Finally, the more specific context of this work, combining all of the above mentioned directions, is *compositional QNLP* – understood as approaches to a quantum implementation of NLP models in the spirit of the DisCoCat model. For an introduction to this model see Section 3.2. The early theoretical work by Zeng and Coecke (2016) leverages the DisCoCat model to obtain a quadratic speedup for a sentence classification task. While, again, the proposed algorithm in that work requires technology that is currently not available, such as 'quantum RAM', subsequent theoretical work (Coecke et al., 2020) lays the foundation for implementations specifically on currently available NISQ devices. Furthermore, in (O'Riordan et al., 2020) a DisCoCat-inspired workflow is introduced along with experimental results obtained by classical simulation. Meichanetzidis et al. (2023) provided for the first time a proof of concept that practical QNLP is in principle possible in the NISQ era, by managing to run and optimize on a NISQ device a classifier using a dataset of 16 artificial short sentences. This current work is a natural next step, presenting for the first time two NLP experiments of medium scale (given the quantum computing context) on quantum hardware.

3. Compositional Models for Sentence Representation

3.1 The General Landscape: From Syntax-Insensitive to Syntax-Sensitive Models

In this section we briefly discuss the topic of sentence modelling and the role syntax plays in it. In the simplest case a sentence can be represented as a "bag of words", that is, an unordered set of symbols or most commonly *embeddings*, that are associated with the words. Since each word is independent of its context, syntactic relationships cannot be modelled in this case. When embeddings or other distributional approaches are used, a simple compositional model for preparing a sentence representation would typically involve element-wise addition or multiplication of the vectors for the words in the sentence (Mitchell & Lapata, 2010). For a few simple NLP tasks, where the examination of the words in isolation is enough to provide the correct result, a bag-of-words model can be a lightweight and effective solution.[4]

However, for most practical real-world tasks, some amount of syntax is usually necessary. A *word-sequence* model respects the order of the words, processing the sentence word by word from left to right. This approach can capture localised interactions between the words and (to some degree) longer-range dependencies as well. The most popular word-sequence model for NLP is the *recurrent neural network* (RNN), with the *long short-term memory* (LSTM) variation (Hochreiter & Schmidhuber, 1997) being the de-facto standard model for NLP-related tasks, at least before the advent of transformers.

Finally, there is a class of models where the order of composition is strictly dictated by the syntactic structure of the sentence. For these cases, a parse tree is provided by a (typically statistical) parser, which for instance, for the sentence "John gave Mary a flower" would be of the following form:

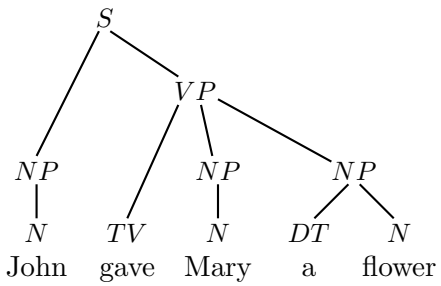


Figure 1: Example of a parse tree with S representing a sentence, VP a verb phrase, NP a noun phrase, TV a transitive verb, N a noun and DT a determiner.

An example of a fully syntax-sensitive approach is the *recursive neural network* (Socher et al., 2012), in which a standard RNN composes the words of a sentence not in sequence from left to right, but by following a provided parse tree. These models have proven very effective in tasks such as sentiment analysis (Socher et al., 2013).

In light of the sketched spectrum of a model's syntax sensitivity, it is important to note that whether fully syntax-sensitive models are useful in NLP or not is still under debate.

4. The meaning classification task we describe in Section 6 is such an example.

In fact, it has been found that in most cases modern neural network (NN) architectures are capable of dynamically learning the syntactic features that are required for the task at hand from the training data on-the-fly, making the need to explicitly provide syntax trees for the sentences obsolete (see, e.g., Iyyer et al., 2015). However, such a degree of learning capacity requires relatively large models and even larger amounts of training data, which is something that – at least currently – cannot be afforded in quantum computing given the status of the available machines. Thus, models intended to be run on quantum hardware have to be chosen wisely.

The DisCoCat model, which is introduced in the subsequent section (Sec. 3.2), is a fully syntax-sensitive model of natural language meaning and is particularly suitable for the purposes of this paper for two reasons. First, it features a particular structural compatibility with quantum theory. This will be explained in detail below, but in short the model's sentence representations can be seen as *tensor networks* (Orús, 2014). This makes the model a natural choice for a quantum implementation and more appropriate than above mentioned syntax-based models from conventional NLP. Second, to address the question of why use a syntax-based model in the first place, we emphasise again the tension between wanting a model that can deal with complex tasks that require syntax, and the impossibility of putting on a quantum computer the kind of large model that learns aspects of syntax from the training data on-the-fly. Hence, the choice of DisCoCat for the experiments of this work brings the additional benefit that a considerable amount of complexity is removed from the training process, since the syntactic structures of the sentences are provided to the model as part of the input.

3.2 A Syntax-Based Model Inspired by Quantum Mechanics

This section introduces and explains in detail the DisCoCat model of Coecke et al. (2010)[5], whose aforementioned structural compatibility with quantum theory is due to the fact that it is based on the rigorous mathematical framework of compact closed categories. Compact closed categories are the structure that also provide an abstraction of the Hilbert space formulation of quantum theory (Abramsky & Coecke, 2004).

In DisCoCat, the meaning of words is represented by tensors whose order is determined by the types of the words according to a *pregroup grammar* (Lambek, 2008). A type p has a left (p^l) and a right adjoint (p^r), and the grammar has only two reduction rules:

$$p \cdot p^r \to 1 \qquad p^l \cdot p \to 1 \tag{1}$$

Assuming atomic types n for nouns and noun phrases and s for sentences, the type of a transitive verb becomes $n^r \cdot s \cdot n^l$, denoting that an n is expected on the left and another one on the right, to return an s. Thus, the derivation for a transitive sentence such as "John likes Mary" is of the form:

$$n \cdot (n^r \cdot s \cdot n^l) \cdot n \quad \to \quad (n \cdot n^r) \cdot s \cdot (n^l \cdot n) \quad \to \quad 1 \cdot s \cdot 1 \quad \to \quad s \tag{2}$$

witnessing that it is a grammatical sentence. This derivation can also be represented diagrammatically as a *pregroup diagram*:

5. The explanation of how one can implement this general, mathematical model for natural language meaning in different ways, on quantum or conventional/classical hardware, is postponed to Sections 5 and 7.

where the "cups" ($\cup$) denote the grammar reductions. The transition from pregroups to vector space semantics is achieved by a mapping[6] $\mathcal{F}$ that sends each atomic type (and its dual types) to a vector space (n to N and s to S) and composite types to corresponding tensor product spaces ($n^r \cdot s \cdot n^l$ to $N \otimes S \otimes N$). For example, a transitive verb becomes a tensor of order 3, which can be seen as a bilinear map $N \otimes N \to S$, while an adjective (with type $n \cdot n^l$) is a matrix, i.e. a linear map $N \to N$. Further, $\mathcal{F}$ translates all grammar reductions to tensor contractions, so that the representation of a sentence $s = w_1 w_2 \dots w_n$ with n words $w_1, w_2, \dots, w_n$ and a pregroup derivation α is given by:

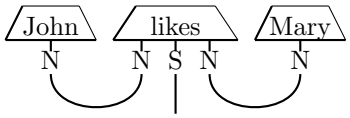
$$\mathbf{s} = \mathcal{F}(\alpha)\left[\mathbf{w_1} \otimes \mathbf{w_2} \otimes \cdots \otimes \mathbf{w_n}\right] \tag{3}$$

where $\mathbf{w_i} = \mathcal{F}(w_i)$, i.e. $\mathbf{w_i}$ is the tensor that the ith word is mapped to, and $\mathcal{F}(\alpha)$, the image of the derivation α under $\mathcal{F}$, is a linear map that when applied to the tensor product of the word representations, by tensor-contracting that expression, returns a vector for the whole sentence. As a concrete example, the meaning of the sentence "John likes Mary" becomes

$$\mathbf{s}_j \;=\; \sum_{i,k} \mathbf{u}_i \; \mathbf{v}_{ijk} \; \mathbf{w}_k \tag{4}$$

where $\mathbf{u}, \mathbf{v} \in N$ are vectors – tensors of order 1 – representing the two nouns with $\mathbf{u}_i$ and $\mathbf{w}_k$ being their respective components with respect to a fixed basis. Similarly, $\mathbf{v} \in N \otimes S \otimes N$ is a tensor of order 3 representing the verb with components $\mathbf{v}_{ijk}$, and $\mathbf{s}$ is a vector in S with components $\mathbf{s}_j$ representing the sentence . Note that the underlying field of the vector spaces is not specified, and can, e.g., be $\mathbb{R}$ or $\mathbb{C}$ depending on the particular type of model. In this work the underlying field of the vector spaces is $\mathbb{C}$.

Meaning computations like the example above can be conveniently represented using the diagrammatic calculus of compact closed categories (Coecke & Kissinger, 2017):



where the boxes denote tensors, the order of which is determined by the number of their wires, and the cups are now the tensor contractions. Note the similarity between the pregroup diagram above with the one here, and how the grammatical derivation essentially dictates both the shapes of the tensors and the contractions. As we will see later, these *string diagrams* can further naturally be mapped to quantum circuits for use in quantum computation.

There is a rich literature on the DisCoCat model, and here we mention only a few indicative publications. Proposals for modelling verbs and implementations of the model have

6. This mapping can be formalised as a category-theoretic *functor* (Kartsaklis, Sadrzadeh, Pulman, & Coecke, 2016).

been provided by Grefenstette and Sadrzadeh (2011) and (Kartsaklis et al., 2012; Kartsaklis & Sadrzadeh, 2014). (Sadrzadeh et al., 2013) address the modelling of relative pronouns. (Piedeleu et al., 2015) present a version of the model where the meaning of words is given by density matrices, encoding phenomena such as homonymy and polysemy. Various versions of the model have been used extensively in conceptual tasks such as textual entailment at the level of sentences, see for example (Sadrzadeh et al., 2018; Bankova et al., 2018; Lewis, 2019). Finally, Yeung and Kartsaklis (2021) reformulate DisCoCat as a passage from a Combinatory Categorial Grammar (CCG) (Steedman, 2001) to a category of semantics, decoupling the model from pregroup grammars and making available to researchers a wide range of CCG resources, such as large-scale collections of human-annotated syntactic trees and statistical parsers.

3.3 Quantum-Friendly Versions of the Word-Sequence and Bag-of-Words Models

As discussed, the DisCoCat model is a syntax-based model that is suitable for quantum implementation. In order to enable a comparative study, and in light of the spectrum of syntax sensitivity sketched in Section 3.1, this section will propose and introduce examples of quantum-friendly versions of the word-sequence and bag-of-words models.

Given the compatibility between tensor networks and the workings of a quantum computer a natural choice for a word-sequence model, simpler than the RNNs mentioned in Section 3.1, is simply to let a sentence representation be a tensor network that reflects the sentence's word order. For the purposes of this work, we propose the simple approach exemplified in Figure 2, in which each word is a tensor of order 2 (a matrix) and their multiplication in the order as the respective words appear in the sentence, defines, once applied to a token "start" vector, the output vector that represents the overall sentence.

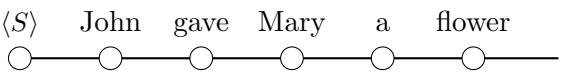


Figure 2: A simple word-sequence model using a tensor network representation. The token $\langle S \rangle$ marks the start of a sentence.

It is instructive to note that this can equivalently be represented as a string diagram, as seen below in Figure 3, by letting cups represent tensor contractions as before (see Section 3.2). The sentence representations in this model are therefore very similar those of DisCoCat. The differences are: a) that here the order of every word's tensor (its arity) is fixed to be the same independent from the words' grammatical types and b) that the connectivity of a diagram (how the tensors are contracted) is not determined by a parse-tree, but simply by the order in which the words appear in the sentence from left to right.

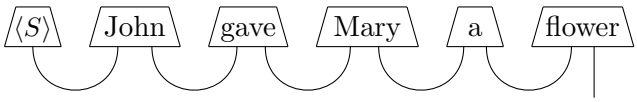


Figure 3: The diagram from Figure 2 rewritten in equivalent form as a string diagram.

Finally, in order to provide a suitable bag-of-words model, again making use of tensor network representations, we use a commutative linear operation that maps a number of vectors to another vector, where it is the commutativity that makes the insensitivity to the word order manifest. A suitable example of such an operation is provided by simple component-wise multiplication given a fixed basis $\{e_1, ..., e_d\}$ of a d-dimensional vector space V. For n input vectors, the map can be defined through its action on the fixed basis as follows

$$\begin{array}{rcl} m \ : \ \bigotimes_{k=1}^{n} V & \rightarrow & V \\ e_{i_1} \otimes \cdots \otimes e_{i_n} & \mapsto & \delta_{i_1 \ldots i_n} \ e_{i_1} \end{array}$$

and, hence, some n vectors, when represented in that fixed basis, are mapped to the vector that is obtained from component-wise multiplication:

$$\Big(\ (v_1^{(1)}, ..., v_d^{(1)}) \otimes ... \otimes \ (v_1^{(n)}, \ ..., v_d^{(n)}) \Big) \ \mapsto \ \Big(\ (\prod_{j=1}^{n} v_1^{(j)}), \ ..., \ (\prod_{j=1}^{n} v_d^{(j)}) \ \Big)$$

This map has a common string diagrammatic representation as a 'merge-dot' as in Figure 4, seeing as it 'merges' the n copies of V into a single copy of V[7]. In this simple model a sentence with n words is thus represented by the vector obtained from applying the multiplication (merge operation) m to the set of n vectors that represent the words of that sentence. Figure 4 depicts the representation of our example sentence with 5 words in this model.

We note that once again the resulting diagrams such as the one in Figure 4, are string diagrams and can be seen as even further simplified versions of DisCoCat representations.

We will refer to these models simply as the word-sequence and bag-of-words models, respectively. We emphasise again that they both yield sentence representations that are simpler versions of the same sort of objects as those of the DisCoCat model, namely string diagrams that can be interpreted in terms of (complex) vector spaces and linear maps between them. This constitutes a compatibility between our models, which will be exploited in Sections 5 and 7, in order to apply a single unified experimental pipeline.

4. Introduction to Quantum Computing

For a serious introduction into quantum information theory and quantum computing, which is beyond the scope of this paper, the reader is referred to the literature (see, e.g., Coecke &

7. More abstractly, the map m is an instance of the multiplication and co-multiplication maps, which are canonically induced by a fixed basis of a vector space, and form a *Frobenius algebra*. See Kartsaklis et al. (2016) for further explanation of how Frobenius multiplication and co-multiplication can be seen as copying and merging the dimensions of a tensor

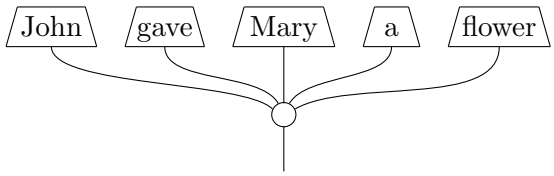


Figure 4: The example sentence in our bag-of-words model. The 'dot' represents the multiplication m, also called 'merge' operation, defined above.

Kissinger, 2017; Nielsen & Chuang, 2011). However, to provide a self-contained manuscript, in this section we will set out the required terms and concepts in such a way that no previous exposure to quantum theory is required.

We start with the concept of a *qubit*, which, as the most basic unit of information carrier, is the quantum analogue of a bit, and yet a very different sort of thing. It is associated with a property of a physical system such as the spin of an electron, which can be 'up' or 'down' along some axis and there is a sense in which a qubit can indeed be in two 'extreme states', corresponding to the two possible outcomes of some measurement, hence the analogy to a classical bit. However, such pairs of extreme, namely perfectly distinguishable states do not exhaust the set of states a qubit can be in – a general *state* $|\psi\rangle$ lives in a 2-dimensional *complex* vector space, more precisely a Hilbert space.[8] In denoting a state vector as $|\psi\rangle$, read 'ket psi', we use notation that is common in physics and called *braket* notation; why decorating a vector with a 'ket' symbol, $|\ \rangle$, is useful will become clear momentarily.

Let $|0\rangle$ and $|1\rangle$ denote orthonormal basis vectors of a qubit's Hilbert space, where the labels of these two states correspond to the respective outcomes of a measurement, which are here generically labelled as '0' and '1'. A general state of the qubit then is a (complex) linear combination known as a *superposition*: $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. A reader not familiar with complex valued linear algebra and Hilbert spaces will be relieved to learn that no such familiarity is necessary to follow the main content of this work, which will instead use a very intuitive diagrammatic language for these structures, introduced below.

Importantly, quantum theory is a *fundamentally probabilistic* theory, that is, even given that a qubit is in state $|\psi\rangle$ – a state known as perfectly as is in principle possible – this generally only allows one to make predictions for the probabilities, with which the outcomes '0' and '1', respectively, occur when the qubit is *measured*. These probabilities are given by the so-called Born rule $P(i) = |\,\langle i|\psi\rangle\,|^2$, where $i = 0, 1$ ranges over the possible outcomes[9] and $\langle i|\psi\rangle$ is a complex number, called the *amplitude*, which is given by the inner product written as the composition of state $|\psi\rangle$ with *bra* vector $\langle i|$. Hence, for the above state $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ the outcome probabilities are given by $P(0) = |\alpha|^2$ and $P(1) = |\beta|^2$.

8. Such states $|\psi\rangle$ are also referred to as *pure* states, as opposed to *mixed* states. The latter are a different kind of object and in particular allow one to represent probabilistic mixtures over pure states when there is a lack of knowledge of which pure state was prepared. The formalism in this work will only make reference to pure states.
9. That is, the integer $i \in \{0, 1\}$ is used to label vectors when writing $|i\rangle$ and $\langle i|$.

We now also see the reason why braket notation is useful. The vector $\langle i|$, which is decorated with a 'bra', $\langle\ |$, denotes the dual vector of $|i\rangle$ in the corresponding dual Hilbert space (basically, $|i\rangle$ can be thought of as a column vector and $\langle i|$ as the corresponding complex conjugated row vector) such that writing the *bra-ket*, $\langle i|\psi\rangle$, gives the inner product of $|\psi\rangle$ and $|i\rangle$ – the very thing that determines probability distributions as the empirical predictions of quantum theory. On a terminological note, a bra $\langle i|$ is in the literature and in this manuscript also referred to as a (quantum) *effect*, reflecting more intuitively that it corresponds to obtaining the measurement outcome i.

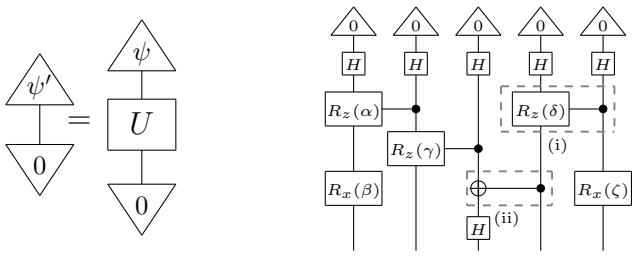
According to quantum theory the evolution of an isolated qubit before measurement, that is the change of its state over time, is described through the *transformation* of its state with a *unitary* linear map U,[10] i.e. the state $|\psi'\rangle$ describing the closed system after such evolution relates to the initial state $|\psi\rangle$ as $|\psi'\rangle = U|\psi\rangle$. The amplitude for obtaining outcome 0, given $|\psi'\rangle$ is $\langle 0|\psi'\rangle = \langle 0|U|\psi\rangle$. The exact same linear equation is represented in Fig. 5a, now starting to introduce the intuitive diagrammatic language for quantum theory and quantum computing promised earlier: the diagrams are read top down; the box labelled U represents the unitary linear map U, while the input and output 'wires' of the box correspond to qubits; triangles without input wires represent states, while triangles that are orientated upside-down compared to those of states and without output wires represent effects – the (non-deterministic) outcomes of measurements.

More generally, the joint state space of q qubits is given by the tensor product of their individual state spaces and thus has (complex) dimension 2^q. For instance, for two 'uncorrelated' qubits in states $|\psi_1\rangle = \alpha_1|0\rangle + \beta_1|1\rangle$ and $|\psi_2\rangle = \alpha_2|0\rangle + \beta_2|1\rangle$, the joint state is $|\psi_1\rangle \otimes |\psi_2\rangle$, which in basis-dependent notation becomes $(\alpha_1, \beta_1)^T \otimes (\alpha_2, \beta_2)^T = (\alpha_1\alpha_2,\ \alpha_1\beta_2,\ \beta_1\alpha_2,\ \beta_1\beta_2)^T$. The evolution of a set of qubits that interact with each other, is then described by a unitary map acting on the overall state space.

The diagrammatic representation from Fig. 5a extends naturally to *quantum circuits* for the general case of several qubits (see, e.g., Coecke & Kissinger, 2017; Coecke & Gogioso, 2022) – the different qubits are represented as parallel wires, and their evolution may now involve boxes with more than one input and output wire seeing as the evolution will generally involve interaction amongst the qubits. An example is shown in Fig. 5b, where 5 qubits are prepared in an initially uncorrelated state of all $|0\rangle$ states and then evolve through a sequence of unitaries. The overall diagram represents the output state, i.e. the state after the evolution happened, at which point the qubits may be measured or may keep evolving further.

The example quantum circuit in Fig. 5b contains all the kinds of gates that make appearance in this paper: the Hadamard gate H is a specific single qubit unitary transformation; similarly, the quantum CNOT gate in (ii) is a specific two-qubit unitary transformation; the gate $Rx(\beta)$, denotes a parameterised unitary, namely for every $\beta \in [0, 2\pi]$ it is an X-rotation by angle β; and finally, there is the controlled Z-rotation gate in (i), a component of which is a Z-rotation gate $Rz(\delta)$ by angle $\delta \in [0, 2\pi]$. Note that knowing the precise definitions of these gates as specific linear maps is irrelevant to understanding the rest of this work, all one needs to know is that the symbols H, $Rx(\beta)$ etc. refer to the sorts of

10. A *unitary* operator U can be represented as a matrix U that satisfies the condition of unitarity, i.e. $U^\dagger U = id = UU^\dagger$, where id is the identity matrix on the vector space and $U^\dagger$ refers to the conjugate transpose of U.



(a) (b)

Figure 5: Basic examples of quantum circuits as a diagrammatic language for quantum theory and quantum computing (see main text for details).

$$H \;:\; |0\rangle \;\mapsto\; \frac{1}{\sqrt{2}}\big(\,|0\rangle + |1\rangle\,\big) \qquad |1\rangle \;\mapsto\; \frac{1}{\sqrt{2}}\big(\,|0\rangle - |1\rangle\,\big)$$

$$Rx(\beta) \;:\; |0\rangle \;\mapsto\; \cos(\frac{\beta}{2})\,|0\rangle - i\sin(\frac{\beta}{2})\,|1\rangle \qquad |1\rangle \;\mapsto\; -i\sin(\frac{\beta}{2})\,|0\rangle + \cos(\frac{\beta}{2})\,|1\rangle$$

$$Rz(\alpha) \;:\; |0\rangle \;\mapsto\; e^{-i\frac{\alpha}{2}}\,|0\rangle \qquad |1\rangle \;\mapsto\; e^{i\frac{\alpha}{2}}\,|1\rangle$$

$$CNOT \;:\; |00\rangle \;\mapsto\; |00\rangle\;,\;\; |10\rangle \;\mapsto\; |11\rangle \qquad |01\rangle \;\mapsto\; |01\rangle\;,\;\; |11\rangle \;\mapsto\; |10\rangle$$

Figure 6: Definition of some basic quantum gates with $\alpha, \beta \in [0, 2\pi]$. The controlled Z-rotation gate in (i) of Fig. 5b is defined analogously to how the quantum CNOT gate features 'controlling', i.e. for the control qubit in $|0\rangle$ the target qubit is unchanged, whereas for $|1\rangle$ a Z-rotation is applied to the target qubit.

maps as just described. For the sake of completeness, however, Fig. 6 gives their definitions in terms of what the respective linear map does to each basis vector.

So on the one hand, a quantum circuit captures the structure of the overall linear map evolving the respective qubits, where parallel 'triangles' and parallel boxes are to be read as tensor products of states and unitary maps, respectively, and sequential 'wiring up' of boxes as composition of linear maps. Hence, a circuit as a whole represents the application of a linear map to a vector, computing the systems' overall state at a later time – in other words, it is simple linear algebra and can be viewed as tensor contraction of complex valued tensors that are represented by the gates. On the other hand, a circuit can conveniently also be seen as a list of abstract instructions for what to actually *implement* on some quantum hardware in order to make, say, some ions[11] undergo, physically, the corresponding transformations as prescribed by the gates of the circuit.

Now, coming back to the fact that quantum theory is probabilistic, once a circuit has been run on hardware all qubits are measured. In the case of Fig. 5b this yields 5 bits each time and many such runs have to be repeated to obtain statistics from which to estimate the outcome probabilities. These probabilities connect theory and experiment. In order to

11. The basic physical object used as a qubit varies vastly across different quantum computers.

obtain the result for a given problem, the design of the circuit has to encode the problem such that that result is a function of the outcome probabilities. Hence, the choice of circuit is key.

A special case worth mentioning due to its relevance for this paper is as follows: the encoding of the quantity of interest in a circuit over, say, q qubits may be such that the result is a function of the outcome distribution on just r of the qubits ($r < q$), but subject to the condition that the remaining $q - r$ qubits have yielded particular outcomes, i.e. the result is a function of the corresponding conditional probability distribution. The technical term for this is *post-selection*, as one has to run the whole circuit many times and measure all qubits to then restrict – post-select – the data for when the condition on the $q-r$ qubits is satisfied. At the diagrammatic level the need for such post-selection is typically indicated by the corresponding quantum effects as done, e.g., in Fig. 10, which up to the 0-effects on 4 of the 5 qubits, is identical to that of Fig. 5b.

Finally and crucially, actually building and running a quantum computer is a challenging engineering task for multiple reasons. Above all, qubits are prone to random errors from their environment and unwanted interactions amongst them. This 'coherent noise' is different in nature to that of a classical computing hardware. A quantum computer that would give the expected advantages for large scale problems, is one that comes with a large number of *fault-tolerant* qubits, essentially obtained by clever error correction techniques. Quantum error correction (Brown et al., 2016) reduces to finding ways for distributively encoding the state of a logical qubit on a large number of physical qubits (hundreds or even thousands). The scalable technical realisability of logical qubits is still out of reach at the time of writing. The currently available quantum devices are rather noisy medium-scale machines with qubit numbers mostly still in the double digit range. These devices provide proof of concept and are extremely valuable assets for the development of both theory and applications. This is the reason one speaks of the NISQ era, mentioned in Sec. 1, and this is the light in which the work in this paper has to be seen – exciting proof of concept, while the machines are still too small and noisy for large-scale QNLP experiments.

5. The General Pipeline

In Section 4 we saw that the quantum computing equivalent to classical programming involves the application of quantum gates on qubits according to a quantum circuit. This section will explain the general pipeline of our approach, and in particular the process of how to go from a sentence to its representation as a quantum circuit, which forms the basis on which the model predicts the label. We first do this explicitly for the DisCoCat model as it constitutes the most intricate of the three models discussed in Section 3. We then make clear how the other two models are subsumed in this pipeline by simply dropping or simplifying particular steps.

Figure 7 schematically depicts this pipeline and in this section we address each numbered step at a generic level. Concrete examples of the choices that one has to make in each step are covered in the implementation of this pipeline presented in Section 7.

Since DisCoCat is fully syntax-sensitive, the first step is to get a syntax tree corresponding to the sentence. From this a DisCoCat derivation is created in a diagrammatic form. In order to avoid computational complications on quantum hardware, this diagram

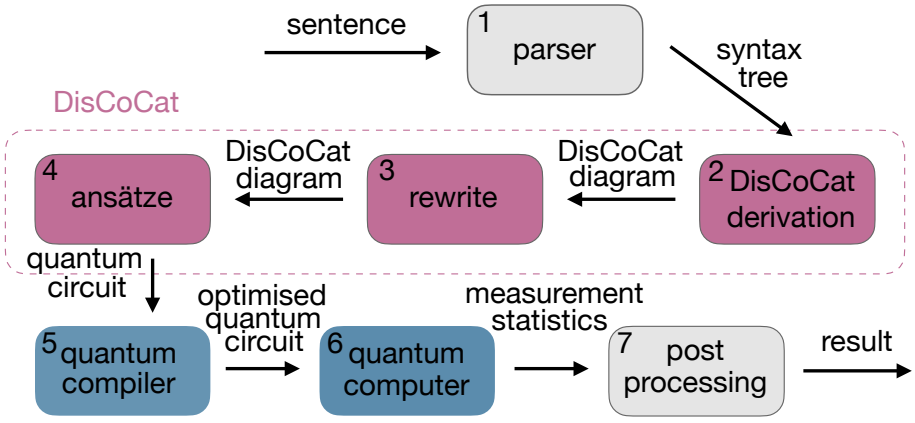


Figure 7: Schematic overview of the general pipeline.

is first optimised to yield the input into an *ansatz* which determines the actual conversion into a quantum circuit. A quantum compiler translates the quantum circuit into hardware-specific code that can be run on quantum hardware. These stages are described in more detail below.

Step 1 (Parser): For a large-scale NLP experiment with thousands of sentences of various structures, the use of a pregroup parser for providing the syntax trees would of course be necessary.[12] In the present work this step can however be executed semi-automatically due to the limited vocabulary and the small number of different grammatical structures in our sentences, that is, once one has worked out the grammatical derivations for all types of sentences or phrases appearing in a dataset, a simple look-up table on the basis of the types of words appearing in the sentence or phrase, allows one to produce the respective parsing. For instance, with nouns, adjectives and transitive verbs having the types n, $n \cdot n^l$ and $n^r \cdot n \cdot n^l$, respectively, the sentence "person prepares tasty dinner" is parsed as below (see Sec. 3.2 for more details on the pregroup grammar and Sec. 7 for details on the specific datasets studied in this work):

$$n \cdot (n^r \cdot s \cdot n^l) \cdot (n \cdot n^l) \cdot n \;\to\; (n \cdot n^r) \cdot s \cdot (n^l \cdot n) \cdot (n^l \cdot n) \;\to\; 1 \cdot s \cdot 1 \cdot 1 \;\to\; s \quad (5)$$

12. Since to the best of our knowledge at the time of writing there are not any robust pregroup parsers available, an alternative approach would be to use a CCG parser and subsequently convert the types into pregroups. Yeung and Kartsaklis (2021) provide the details of a functorial passage from CCG derivations encoded in a biclosed category to DisCoCat diagrams, and a web tool that demonstrates the conversion (`https://qnlp.cambridgequantum.com/generate.html`). Furthermore, the open-source Python package `lambeq` (Kartsaklis et al., 2021) fully automates the conversion of a sentence to a monoidal diagram and then to a quantum circuit by coupling to a CCG parser and enables the user to design and implement compositional QNLP experiments; more information can be found at `https://cqcl.github.io/lambeq`.

Step 2 (DisCoCat Derivation): Construct the sentences' DisCoCat diagrams by representing each word as a state, i.e. a box, and then 'wiring them up' by drawing a cup for every reduction rule. The above example becomes:[13]

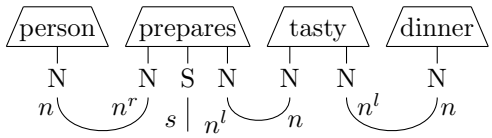


Figure 8

Step 3 (Rewrite): The structure of compact closed categories comes with rewrite rules that allow the transformation of diagrams such as the one shown above into equivalent ones. The reason this is relevant is that a different and yet equivalent string diagram can come with computational advantages when actually running a model. These advantages can be hardware specific – e.g. that a particular gate is more costly to implement than others, or hardware agnostic and due to general facts about quantum information processing. Ideally, one would of course want a comprehensive optimisation algorithm that, when given a model, a task and a chosen quantum hardware finds the most efficient string diagram. However, such does not exist yet and, in any case given how recent the field is, it is too early to tell what the kinds of hardware and models will be in future that the algorithm should range over.

The purpose of this exposition here is to make two points. First, for any given model and experimental implementation, the pipeline we are presenting will typically involve some rewriting at this stage for said reasons. Second, for the particular models this work studies there is indeed an obvious and very useful rewrite procedure. We will make a start with explaining it here, as this is where in the pipeline it is taking place, while a full understanding of why this method makes sense will built up through the discussion of the remainder of the pipeline below.

Our specific rewrite method has to do with two facts: cups are costly to implement – basically because they involve obtaining particular measurement outcomes on several qubits that only occur non-deterministically (see Steps 6 and 7 for more details) – but there is a simple, and yet effective equivalence transformation on our string diagrams that reduces the number of cups. The latter is achieved by 'bending down' all nouns of a sentence, by which we mean that, for instance, the string diagram from Fig. 8 is mapped to the following diagram:

13. As discussed in Sec. 3.2, the diagram in Fig. 8 represents linear-algebraic operations between tensors in vector spaces N, S and tensor products of them. For convenience, we also include the pregroup types as a reminder of the grammatical rules involved in each contraction. For the remainder of this paper, only the pregroup types will be shown since they are indicative of the vector spaces they are mapped to.

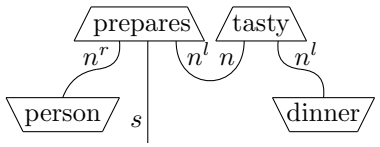


Figure 9

Here the states for 'person' and 'dinner' have been 'bent' down, which reduced the overall number of cups in the string diagram from three to a single one. That the diagrams really are representing the exactly same mathematical object is basically down to the structure of *compact closure*. While the formal details behind it can be read up, e.g., in (Coecke & Kissinger, 2017), the relevant core fact here is intuitive and easy to understand. Compact closure has to do with the presence of the cups, which at the level of the pregroup grammar represent type reductions (see Sec. 3.2). The above equivalence transformation on the diagram bends a word box down and represents it with a box that – recalling diagrams are read top down – has an ingoing wire but no outgoing one. This is in contrast to the respective original word boxes from Fig. 8, which have no ingoing wire, but an outgoing one. That such rewriting is allowed is due to a 1-to-1 correspondence between these two representations of words. This correspondence can be seen to be established by simply defining the corresponding 'turned around' box as the composition of the original noun with a cup as follows:

This correspondence does not only exist at the level of the pregroup grammar, but also at the level of the (complex) vector spaces that the pregroup types then get mapped to – here this correspondence can be thought of as the isomorphism between column vectors and row vectors through the operation of taking the (conjugate) transpose.

Above rewrite procedure thus generally reduces the number of cups by as many nouns as are present in the sentence.

Step 4 (Ansätze): In this step a sentence's abstract DisCoCat representation takes a more concrete form; its DisCoCat diagram is mapped to a specific parametrised quantum circuit. This map is determined by choosing:

(a) the number q_n and q_s of qubits that every wire of type n and s, respectively, as well as their dual types, get mapped to;

(b) concrete parametrised quantum states (effects) that all word states (effects) get consistently replaced with.

We refer to the conjunction of such choices as an *ansatz*. Principled approaches to choosing (b) are presented in Sec. 7, but for an illustration consider again the example from Fig. 9 translated into a parametrised quantum circuit of the form as shown in Fig. 10.

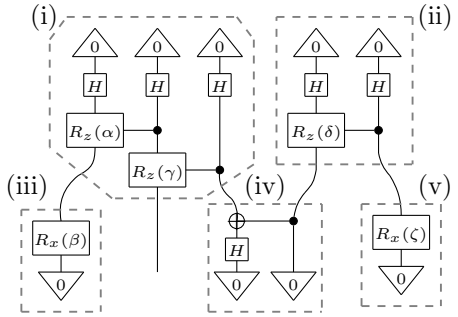


Figure 10: Example of interpreting Fig. 9 as a quantum circuit according to an ansatz. Here $q_n = 1 = q_s$, i.e. one qubit per sentence and noun type; the words 'prepares' and 'tasty' are replaced with the parametrised quantum states as marked by (i) and (ii), respectively; the words 'person' and 'dinner' are replaced with the parametrised quantum effects as marked by (iii) and (v), respectively; the component (iv) is a concrete representation of the cup in quantum circuits (NB an unparametrised quantum effect).

Once an ansatz is chosen such as the one sketched in Fig. 10, a concrete embedding of some word is thus fixed by further fixing particular values for the parameters that appear in the respective word's parametrised quantum state (effect). For instance, the word 'tasty' from Fig. 9 is mapped to the component (ii) of Fig. 10, which for every $\delta \in [0, 2\pi]$ prepares a specific two-qubit quantum state. Note that under this functorial mapping according to some ansatz any cup[14] in a DisCoCat diagram has an already fixed meaning, it is a particular quantum effect,[15] which can be represented as done in the component (iv) of Fig. 10.

It is worth emphasising that the output of the mapping is a parametrised quantum circuit whose connectivity is fixed by the sentence's syntax, while the choice of ansatz determines the number of parameters for each word's representation.

Now, in principle it is of course known how many parameters p are needed to fix the most general state on q qubits, so why does this choice of ansatz matter (independent from questions of overfitting and generalisation)? There are two reasons, which are both of a practical nature. First, p is exponential in q. So, even beyond the NISQ era, for the sort of dataset sizes and lengths of sentences one wishes to consider in NLP, a feasible number of parameters has to be achieved. This is to say, in practice one will rarely afford to work with a fully general parametrised quantum state that can 'hit' any state in some multi-qubit state space for some choice of the parameters. Second, different quantum machines have different sets of 'native' gates, and some gates are less prone to errors than others when implemented. Hence, on NISQ machines the choice of ansatz should be informed by the actual hardware to be used, to avoid unnecessary gate-depth after compilation and hence noise from mere re-parametrisation.

Step 5 (Quantum Compiler): A quantum compiler translates the quantum circuit into machine specific instructions. This includes expressing the circuit's quantum gates in terms of those actually physically available on that machine and 'routing' the qubits to allow for

14. Recall from the discussion in Sec. 3.2 that cups can also be seen to correspond to tensor contractions.
15. In the physics literature also known as a *Bell-effect*.

the necessary interactions given the device's topology, as well as an optimisation to further reduce noise.

Step 6 (Quantum Computer): The quantum computer runs the circuit. In order to see what this means more precisely, recall that the 0-effects in Fig. 10, while crucial parts of the sentence's representation, are not deterministically implementable operations; as outcomes of measurements, they are obtained only with a certain probability. The circuit, which corresponds to that of Fig. 10, but that can actually be implemented, is hence precisely that of Fig. 5b with the additional operation of measuring all 5 qubits at the end.

Thus the quantum computer runs a given circuit n_{shots} times (runs also referred to as *shots*), but ignoring the 0-effects that appear in a sentence's representation for the time being. For each run the machine prepares initial states, applies the gates and then measures all the qubits. At the end, this step returns outcome counts of the shots for *all* qubits.

Step 7 (Post-Processing): The post-processing step of the pipeline has two parts to it. The first is a task independent one and is the completion of what it means to implement a sentence's circuit representation like the one in Fig. 10, that is, taking into account where there are 0-effects (or more generally other non-deterministic effects).

This thus is the point at which *post-selection* happens, a concept that was introduced in Sec. 4. As part of *Step 6* all qubits were measured yielding counts of outcomes. Post-selection now means to restrict the measurement data to the subset of runs where all those qubits with a 0-effect on them in the circuit to be achieved, did indeed yield the corresponding 0 outcome when measured. Only once the data has been appropriately restricted in such manner are the counts turned into estimations of relative frequencies for the outcomes of those qubits that correspond to the 'open' wires in the run circuit, such as the wire coming out of component (i) in Fig. 10.

So the business of post-selection is a simple matter of conditioning! However, we now finally see the reason for why cups are costly: each cup leads to $2q_n$ (or $2q_s$) qubits that require post-selection. Had one stuck to the diagram in Fig. 8 then 6 out of 7 qubits would have had to be post-selected, rather than 4 out of 5 based on Fig. 9. If all possible outcomes were uniformly distributed, even just post-selecting 4 qubits to yield the outcome 0, would mean that only one in 2^4 shots meets the condition. Depending on the actual values of the parameters for which a parametrised circuit is run, if post-selecting on p qubits the probability for seeing the p qubits to yield outcome 0 can be significantly lower (or larger of course) than $1/(2^p)$ and will vary as a training algorithm varies the parameters. Hence, with sentences slightly longer than in our running example and limited number of shots[16] of any given circuit one easily runs into severe statistical limitations to reliably estimate the desired outcome probabilities. The method of 'bending down nouns' (*Step 4*) therefore is an effective and useful rewrite procedure for the models this work studies.

16. For IBM quantum devices the maximum number of shots was 2^{13} at the time of running this experiment. Then post-selecting 6 out of 7 (4 out of 5) qubits and assuming a uniform distribution means that 2^7 (2^9) runs give counts over the remaining qubit whose outcome distribution one wants to estimate. Just to reiterate: for any specific choice of parameters in the circuit the distribution is likely not to be uniform and the runs that meet the conditions can be many fewer or more. Generically, nothing much can be said; we only note that in our experimental set-up explained in detail in Sec. 7 post-selecting 6 qubits sometimes would leave only very few (of the order of 10) useable shots, whereas post-selecting only 4 always yielded sufficient statistics – hence, the presented rewrite procedure does the job for the purpose of this work.

Lastly, the second part of this *Step 7* takes the relative frequencies as input and completes any further post-processing on them for the calculation of a *task-specific* final result such as the computation of a cost function.

5.1 The Pipeline Adjusted to the Word-Sequence Model

The pipeline described above also applies to the word-sequence model with the following simplifications.

Step 1 is irrelevant since no parse tree is input into this model.

In *Step 2*, instead of a DisCoCat diagram based on a parse-tree, a string diagram of the form shown in Figure 3 is created. Following a state with a single output wire for the 'start' token on the very left, every word is represented by a state with two output wires and then every adjacent pair of states is connected up with a cup. For the example sentence of this section the corresponding diagram is shown in Figure 11a. Note that the wires have no explicit typing that would reflect the words' composite grammatical types like in the syntax-based DisCoCat model. However, in light of the intended interpretation of the diagram in the category of (complex) vector spaces, all wires are to be treated as of the same kind. For the remainder of the pipeline to go through the wires thus have to be typed consistently with, say, n – it does not matter what the label is as long as it is the same for all of them.

Concerning *Step 3*, since this work will only present classical simulations for this model rather than an actual quantum implementation, there is no practical need to fix any particular rewrite procedure to optimise the string diagrams. We only note that this could easily be done.[17]

Finally, concerning *Steps 4-7*, these apply in the same way as for DisCoCat. In particular, the same choices constituting an ansatz for the DisCoCat model in *Step 4* specify a map that turns any word-sequence string diagram into a specific quantum circuit. (NB the above comment on treating all wires as labelled consistently with, say, n).

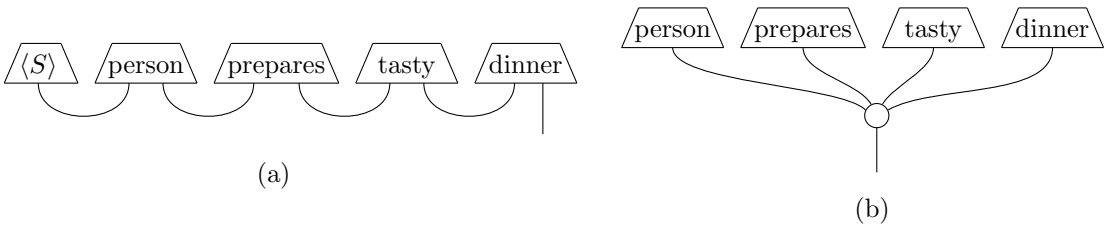


Figure 11: The string diagram representation of step 2 for the example sentence according to the word-sequence model in (a) and according to the bag-of-words model in (b), respectively.

While the pipeline thus is general and can instantiate the word-sequence model, too, we note that for this baseline model we will present experimental results from classical simulations (see Sec. 7.3), but not experiments from an implementation on a quantum computer. See Sec. 7.5 for the reasoning behind this choice.

17. A suitable choice would e.g. be to start from the right of the diagram and flip every other state into an effect according to the *bigraph* recipe from (Meichanetzidis et al., 2021).

5.2 The Pipeline Adjusted to the Bag-of-Words Model

The same simplifications of the pipeline laid out for the word-sequence model, also apply for the bag-of-words model. The only difference is that in *Step 2* the construction of the string diagram for a sentence follows the recipe for this model: every word is represented by a state with a single output wire which are all connected with a single 'merge-dot'. Figure 11b depicts the resulting diagram in the bag-of-words model for the example sentence. Note that when specifying an ansatz in *Step 4*, just as a cup has a fixed meaning in the resultant quantum circuit (see the discussion of *Step 4* in the general pipeline above), so does the merge-dot have a fixed meaning (given a fixed basis of the underlying vector spaces). See Sec. 3.3 for the definition of the map represented by the merge-dot and see Section 7 for details on how to realise this map explicitly through a choice of quantum gates.

Again, this establishes the generality of the pipeline, allowing us to treat all three kinds of models in a unified way, but just as for the word-sequence model, also for this baseline model we will present only experimental results from classical simulations.

6. The Tasks

We define two simple binary classification tasks for sentences. In the first one, we generated sentences of simple syntactical forms (containing 3-4 words) from a fixed vocabulary by using a simple context-free grammar (Table 1). The nature of the vocabulary (of size 17) allowed

`noun_phrase` → `noun`
`noun_phrase` → `adjective noun`
`verb_phrase` → `verb noun_phrase`
`sentence` → `noun_phrase verb_phrase`

Table 1: Context-free grammar for the *MC* task.

us to choose sentences that look natural and refer to one of two possible topics, food or IT. The chosen dataset of this task, henceforth referred to as *MC* ('meaning classification'), consists of 65 sentences from each topic, similar to the following:

"skillful programmer creates software"
"chef prepares delicious meal"

Part of the vocabulary (four words) is shared between the two classes, so the task (while still an easy one from an NLP perspective) is not trivial.

In a slightly more conceptual task, we select 105 noun phrases containing relative clauses from the RelPron dataset (Rimell et al., 2016). The phrases are selected in such a way that each word occurs at least 3 times in the dataset, yielding an overall vocabulary of 115 words. While the original task is to map textual definitions (such as "device that detects planets") to terms ("telescope"), for the purposes of this work we convert the problem into a binary-classification one, with the goal to predict whether a certain noun phrase contains a subject relative clause ("device that detects planets") or an object relative clause ("device that observatory has"). Our motivation behind this task, henceforth referred to as *RP*('relative pronoun'), is that it requires some syntax sensitivity from the model, so it is a reasonable choice for testing the DisCoCat model. In addition, the size of vocabulary and

consequently the sparseness of words make this task a much more challenging benchmark compared to the *MC* task.

These simple datasets already pose challenges in two ways. First, concerning the lengths of sentences (see *Step 5* of Sec. 5 and also Sec. 8). Second, concerning the lengths of datasets, since they already reach the limits of the currently available quantum hardware – even just doubling the number of sentences would start to approach an unfeasible time cost given the shared available resources (more details about this in Sec. 7.5).

7. Experiments

The experiments reported in this paper address the tasks *MC* and *RP* by implementing the pipeline from Fig. 7 together with an optimisation procedure to train the model parameters against an objective function, as in standard supervised machine learning. Importantly, for all models that this work discusses, training the model amounts to learning the word representations in a task-specific way. Recall that the specificities of these word representations – the size of the embedding space and how the parameters define the word representations – are contingent on the choice of hyperparameters that fix the ansatz, while all other aspects of the sentence or phrase representations are dictated by syntax (to the degree that the model accounts for syntax at all).

As explained in Section 3.1 the DisCoCat model is the most involved, expressive and interesting model for the purpose of this paper. The bag-of-words and the word-sequence models take the role of baseline models in order to show through a comparison that, given the different natures of the *MC* and *RP* tasks, the three distinct models' performances and behaviours are as expected and well understood. All models are simulated and in addition the DisCoCat model is implemented on real quantum hardware.

Our Python implementation[18] used `lambeq`[19] and the DisCoPy package[20] (de Felice et al., 2020) to implement the model specific *Steps 2-4*, the Python interface of the quantum compiler TKET$^{\text{TM}}$ [21] (Sivarajah et al., 2020) for *Step 5*, and the IBM device `ibmq_bogota` for *Step 6*. The remainder of this section describes all steps in detail.

For the first step of parsing sentences, necessary for the DisCoCat model, Section 5 noted that due to the simplicity of both tasks' datasets this parsing can actually be done semi-automatically. For the noun phrases and sentences considered in this work note that relative pronouns in the subject case take pregroup type $n^r \cdot n \cdot s^l \cdot n$, while in the object case their type is $n^r \cdot n \cdot (n^l)^l \cdot s^l$, and that the types for adjectives and transitive verbs are $n \cdot n^l$ and $n^r \cdot s \cdot n^l$, respectively. These are the only pregroup types needed across both our datasets, in order to work out the derivations (see Sec. 3.2 and *Step 2* in Sec. 5) – one can easily convince oneself that all phrases and sentences in our datasets then have a unique reduction to n or s, respectively. For an example of how to implement the parsing step in a semi-automatic way simply using a look-up table with the very few different sentence structures appearing in the *MC* and *RP* datasets, see the github repository in footnote 19.

18. The Python code and the datasets are available at `https://github.com/CQCL/qnlp_lorenz_etal_2021_resources`.
19. `https://cqcl.github.io/lambeq`
20. `https://github.com/discopy`
21. `https://github.com/CQCL/pytket`

In order to illustrate this point, note that the DisCoCat diagrams in Figures 12a and 12b depict two noun phrases from the *RP* dataset. The corresponding representations according to the baseline models for the first of these two examples can be found in Figures 12c and 12d. Furthermore, the example sentence that was used in Section 5 is in fact from the *MC* dataset – Figure 8 showed it for the DisCoCat model and Figure 11 for the baseline models. The scheme of 'bending the nouns', described in *Step 3* of Section 5, was consistently applied to both datasets in case of the DisCoCat model, while skipped for the baseline models (see Section 5 for an explanation).

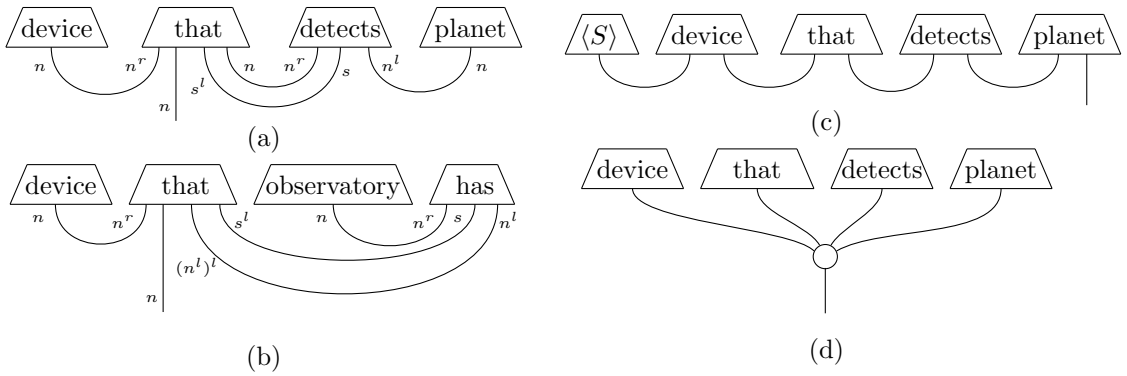


Figure 12: DisCoCat diagrams for example phrases from the RelPron dataset (Rimell et al., 2016), used in the *RP* task, where in (a) 'device' is the subject of the verb, while in (b) it is the object; (c) and (d) show the corresponding diagrams for the sentence from (a), but in the word-sequence and the bag-of-words models, respectively.

As emphasised in Section 5, after the first three (model specific) steps the remaining steps of the pipeline are the same for each model. After *Step 3* the sentences (or noun phrases) for all models are now represented by string diagrams, waiting to be translated into quantum circuits. As mentioned previously, it is convenient – as a mere matter of bookkeeping – to assign the type n to all wires of a diagram in the bag-of-words and the word-sequence models independently from whether the diagram represents a sentence or a noun phrase. This allows for the same hyperparameters that determine an ansatz for the DisCoCat model to also determine the ansatz for the baseline models in a way that ensures the number of parameters to be comparable.

7.1 Model Parametrisation and Ansätze

The exposition of *Step 4* in Section 5 explained how the choice of ansatz determines the parametrisation of a concrete family of models. We studied a variety of ansätze for both tasks. For each ansatz all appearing parameters are valued in $[0, 2\pi]$.

In order to keep the number of qubits as low as possible in light of the noise in NISQ devices, while having at least one qubit to encode the label (we are trying to solve a binary classification task after all) we set $q_n = 1$ and $q_s = 1$ for the *MC* task, but $q_s = 0$ for the *RP* task (noting that the type of the phrases here is n for all three models). Recall that the vector spaces in which the noun and sentence embeddings live have (complex) dimension $N = 2^{q_n}$ and $S = 2^{q_s}$, respectively. Once the dimensions of the vector spaces that the wires

represent are thus fixed, a principled way has to be provided for how to assign concrete (parametrised) quantum states and effects to all appearing word boxes.

For single qubit states $|w\rangle$ (and their corresponding effects $\langle w|$) two options were considered. First, an *Euler parametrisation*, $|w\rangle = Rx(\theta_3)Rz(\theta_2)Rx(\theta_1)\,|0\rangle$. The name is owed to the well-known Euler parametrisation of a general rotation in 3-dimensional space ($\mathbb{R}^3$), but all that matters here is that it is a choice of a fully general parametrisation of any single qubit state and that, as a sequence of three rotation gates, it hence involves three parameters that are angles, $\theta_1, \theta_2, \theta_3 \in [0, 2\pi]$. See Fig. 13a for the corresponding diagrammatic representation of such an Euler parametrisation of a single qubit state, as well as Sec. 4 and in particular Fig. 6 for the details on the involved gates. Second, the use of a single Rx gate by assigning $|w\rangle = Rx(\theta)\,|0\rangle$, which, with a single parameter, gives a more economical option that is still well-motivated, since an Rx gate can mediate between the $|0\rangle$ and $|1\rangle$ basis states. Let $p_n \in \{3, 1\}$ represent the choice of which of these two options is chosen. Any chosen of the two options is then always consistently applied to all words of the dataset with a single qubit representation. Note that for the DisCoCat model, due to our scheme from *Step 3* (see Sec. 5), all nouns appear as single qubit effects $\langle w|$ and for the bag-of-words model all words are single qubit states, while for the word-sequence model only the start token $\langle S\rangle$ is assigned a single qubit state.

In the DisCoCat model adjectives are states on two qubits and verbs (only transitive ones appear) are, depending on q_s, states on two or three qubits, whereas in the word-sequence model all words are two qubit states. For such multi-qubit states so called IQP[22]-based states were used. For m qubits such a state consists of all m qubits initialised in $|0\rangle$, followed by d many IQP layers (Havlíček et al., 2019). Each such layer in turn consists of an H gate on every qubit, subsequently composed with $m-1$ controlled Rz gates, connecting adjacent qubits. For examples see components (i) and (ii) of Fig. 10, which show one IQP layer for a three and a two qubit state, respectively. We stress that the particular choice for the multi qubit state part of an ansatz does not deserve too much significance. One could have chosen some different state parametrisation, i.e. some different well-defined recipe of parametrised layers of gates so long as it satisfies some basic desiderata. The latter are that the qubits can actually interact (at least for most values of the parameters) as is the case in IQP layers through the controlled Rz gates, and that as the number d of layers increases one can (asymptotically) reach any state on the m qubits. The particular choice of IQP layers then is, apart from that it satisfies these features, motivated by that it has been studied in the literature, was found to be expressible enough to perform QML (Havlíček et al., 2019), is such that the appearing gates were native to IBMQ's machines at the time the experiments were run and finally, simply because it worked. We considered $d \in \{1, 2\}$, again in order to keep the depth of circuits as small as possible.

For the RP task, the relative pronoun 'that' also appears in the vocabulary. For the baseline models this relative pronoun is treated like any other word, for the DisCoCat model it however receives a special treatment. Note that although at the pregroup level the type of 'that' depends on whether it is the subject or object case, at the quantum level, recalling that $q_s = 0$ for this task, only one kind of quantum state is required. Following the use of the merge-dot in (Sadrzadeh et al., 2013, 2014)[23] to model functional words like relative

22. Instantaneous Quantum Polynomial.

23. As mentioned in Sec. 3.3 the merge-dot corresponds to the presence of a Frobenius algebra.

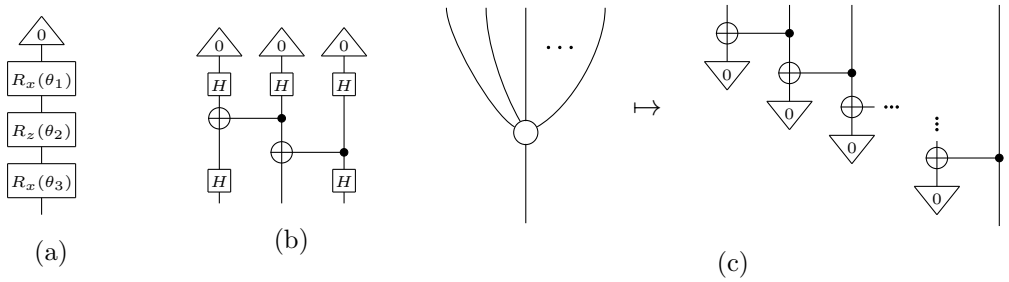


Figure 13: The diagram in (a) shows the Euler parametrisation of a general single qubit state; (b) shows the quantum state (a GHZ state) assigned to 'that' as part of our ansätze in case of the DisCoCat model and given that $q_n = 1$ and $q_s = 0$; (b) depicts the quantum circuit implementation of the 'merge-dot' as part of the bag-of-words model given that all words are assigned single qubit states.

pronouns, we chose for 'that' a GHZ-state, which is displayed in Figure 13b and which, notably, does not involve any parameters.

Finally, the use of the merge-dot also appeared in Section 3.3 in the definition of our bag-of-words model. Figure 13c shows the implementation of that dot as part of a quantum circuit – again, an unparametrised structure.

With the laid out approach, the choices that fix an ansatz (with $q_n = 1$ fixed) can be summarised by a triple of hyperparameters (q_s, p_n, d). The total number k of parameters, denoted $\Theta = (\theta_1, ..., \theta_k)$, varies correspondingly with the model and depends on the vocabulary. See Tables 2 and 3 for the ansätze we studied. Note that Fig. 10 shows the quantum circuit for the example sentence from the *MC* task precisely for ansatz $(1, 1, 1)$ in the DisCoCat model.

MC		*RP*	
(q_s, p_n, d)	k_D	(q_s, p_n, d)	k_D
$(1, 1, 1)$	22	$(0, 1, 1)$	114
$(1, 1, 2)$	35	$(0, 1, 2)$	168
$(1, 3, 1)$	40	$(0, 3, 1)$	234
$(1, 3, 2)$	53	$(0, 3, 2)$	288

Table 2: Overview of the ansätze studied for the DisCoCat model, where k_D is the number of parameters of the resultant model.

MC			*RP*		
(q_s, p_n, d)	k_W	k_B	(q_s, p_n, d)	k_W	k_B
$(1, 2, 1)$	-	34	$(0, 1, 1)$	116	-
$(1, 3, 1)$	-	51	$(0, 1, 2)$	231	-
$(1, 3, 2)$	37	-	$(0, 2, 2)$	-	230

Table 3: Overview of the ansätze studied for the word-sequence and bag-of-words models, where k_W, k_B are the number of parameters for the respective resultant models. Note that for the baseline models q_s is irrelevant, since the only wire type that appears is n (see Secs. 5.1 and 5.2), however it is still included to have a consistent notation.

7.2 Model Prediction and Optimisation

Let P denote a sentence in case of the *MC* dataset or a noun phrase in case of the *RP* dataset. After *Step 4* of the pipeline, every such P is represented by a quantum circuit according to the chosen model and ansatz. Let the corresponding output quantum state,[24] parametrised by the set of parameters Θ, be denoted $|P(\Theta)\rangle$. Given the models and hyper-parameters studied in this work (see Tables 2 and 3) $|P(\Theta)\rangle$ always is a single qubit state, i.e. a vector in a two-dimensional complex Hilbert space. We define

$$l^i_\Theta(P) \;:=\; \Big|\; |\langle i|P(\Theta)\rangle|^2 - \epsilon \;\Big| \tag{6}$$

where $i \in \{0, 1\}$ and ϵ is a small positive number, in our case set to $\epsilon = 10^{-9}$, which ensures that $0 < l^i_\Theta(P) < 1$ and $l^0_\Theta(P) + l^1_\Theta(P) < 1$, so that

$$l_\Theta(P) := \frac{1}{l^0_\Theta(P) + l^1_\Theta(P)} \left(l^0_\Theta(P),\; l^1_\Theta(P)\right) \tag{7}$$

defines a probability distribution. The label for P as predicted by the model is then obtained from rounding, i.e. defined to be $L_\Theta(P) := \lceil l_\Theta(P) \rceil$ with $[0, 1]$ ($[1, 0]$) corresponding to 'food' ('IT') for the *MC* task and to the 'subject case' ('object case') for the *RP* task.

The *MC* dataset was partitioned randomly into subsets $\mathcal{T}$ (training), $\mathcal{D}$ (development) and $\mathcal{P}$ (testing) with cardinalities $|\mathcal{T}| = 70$, $|\mathcal{D}| = 30$, $|\mathcal{P}| = 30$. Similarly, for the *RP* task with $|\mathcal{T}| = 74$, $|\mathcal{P}| = 31$, but no development set $\mathcal{D}$, since the ratio of the sizes of vocabulary and dataset did not allow for yet fewer training data, while the overall dataset of 105 phrases could not be easily changed[25]. For the *RP* task the ratio between subject to object cases was 46/28 in the training subset and 19/12 in the test subset, which reflects the ratio between the two classes in the given overall dataset. For the *MC* task the ratio between 'food' and 'IT' related sentences was 39/31 in the training subset, 11/19 in the development subset and 15/15 in the test subset.

The objective function used for the training is standard cross-entropy; that is, if letting $L(P)$ denote the actual label according to the data, the cost is

24. Generally, a sub-normalised state (in physics jargon).

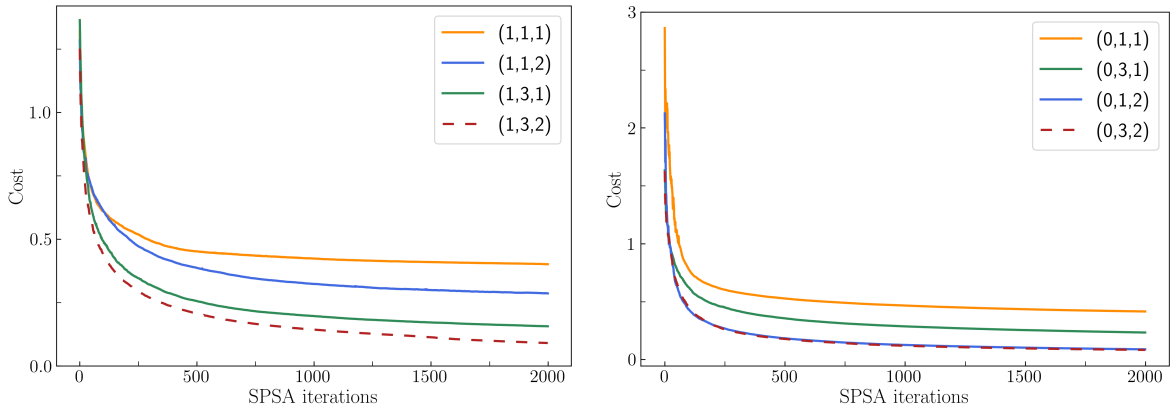
25. In contrast to the *MC* task, here the data was extracted from an existing dataset, and picking further phrases while ensuring a minimum frequency of all words was non-trivial (see Sec. 6).

$$C(\Theta) := -\frac{1}{|\mathcal{T}|}\sum_{P\in\mathcal{T}} L(P)^T \cdot \log\big(l_\Theta(P)\big) \tag{8}$$

For the minimisation of $C(\Theta)$, the SPSA algorithm (Spall, 1998) is used, which for an approximation of the gradient uses two evaluations of the cost function (in a random direction in parameter space and its opposite). The reason for this choice is that in a variational quantum circuit context like here, proper back-propagation requires some form of 'circuit differentiation' that would in turn have to be evaluated on a quantum computer – something being actively developed but still unfeasible from a practical perspective. The SPSA approach provides a less effective but acceptable choice for the purposes of these experiments. Finally, no regularisation was used in any form.

7.3 Classical Simulation

Owing to the fact that computation with NISQ devices is slow, noisy and limited at the time of writing, it is not practical to do extensive training and comparative analyses on them. This was instead done by using classical calculations to replace *Steps 5-6* of Fig. 7 in the following sense. For any choice of parameters Θ and some sentence or phrase P, the complex vector $|P(\Theta)\rangle$ can be calculated by simple linear algebra – basically through tensor contraction. Hence the values $l_\Theta(P)$, and thereby also the cost $C(\Theta)$ as well as the respective types of errors, can be obtained through a 'classical simulation' of the pipeline.



(a) (b)

Figure 14: Convergence of the DisCoCat models in the classical simulation (averaged over 20 runs) for different ansätze; in (a) for the *MC* task and in (b) for the *RP* task.

We therefore use this way of classically simulating the pipeline in particular to compare the different kinds of models that were introduced in preceding sections, as well as to compare the different ansätze for a fixed kind of model. Given that DisCoCat is the most complex of our models and also the main model to be studied in the actual quantum implementation, we start by comparing the different DisCoCat models that result from the different ansätze with corresponding hyperparameters as listed in Tab. 2.

Figs. 14a and 14b present the convergence on the training datasets, of the DisCoCat models for the *MC* and *RP* task, respectively, for the selected sets of ansätze. Shown is the cost over SPSA iterations, where each line is the result of averaging over 20 runs of the optimisation with a random initial parameter point Θ. The reason for this averaging is that there are considerable variances and fluctuations between any individual run due to the crude approximation of the gradient used in the stochastic SPSA algorithm and the specificities of the cost-parameter landscape. As is clear from the plots, the training converges well in all cases. What is more, the dependence of the minima that the average cost converges to, on the chosen ansatz reflects the theoretical understanding as follows.

For the *MC* task, the minimum is the lower the more parameters the model has. For the *RP* task – being about syntactic structure, essentially about word order – the situation is different but in a way that again is understandable. Given our treatment of 'that' (cf. Sec. 7.2), it is not hard to see that the task comes down to learning embeddings such that the verbs' states become sensitive to which of their two wires connects to the first and the second noun in the phrase. Hence, the larger d (which fixes the number of parameters for verbs) the lower the minimum.

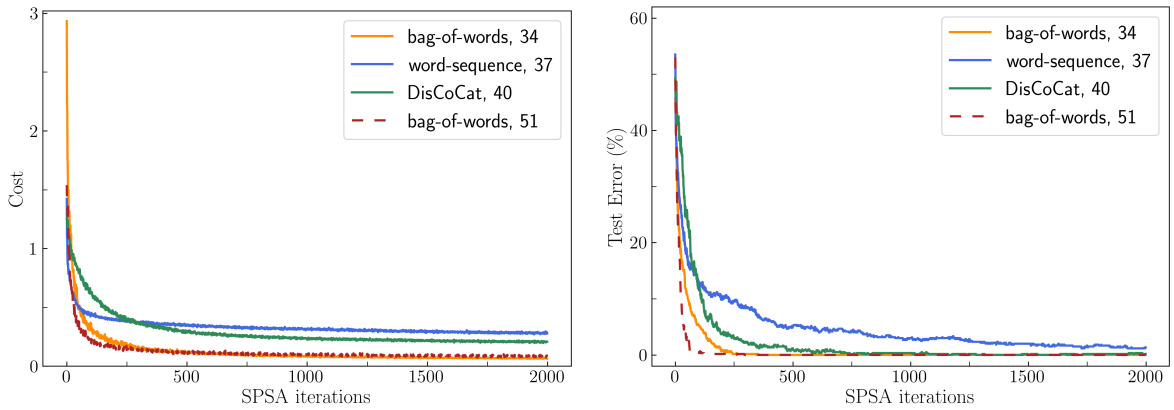
The plots in Figs. 14a and 14b showcase what is expected from a quantum device if it were noise-free, and if many iterations and runs were feasible time-wise. On that basis, we chose one DisCoCat ansatz per task, for implementation on quantum hardware, that does well with as few parameters as possible: $(1, 3, 1)$ for the *MC* task and $(0, 1, 2)$ for the *RP* task.

It is these two selected models which we therefore focus on for the comparison with the simpler baseline models. For the *MC* task we choose to compare the (1,3,1) DisCoCat model (40 parameters) with the (1,2,1) and (1,3,1) bag-of-words models (with 34 and 51 parameters, respectively) and with the (1,3,2) word-sequence model (37 parameters). The reason for these choices is that we wish to keep a similar number of parameters compared to the DisCoCat model for a fair comparison.

For similar reasons we choose to compare in the *RP* task the chosen (0,1,2) DisCoCat model (168 parameters) with the (0,2,2) bag-of-words model (230 parameters) and the (0,1,1) and (0,1,2) word-sequence models (with 116 and 231 parameters, respectively.)

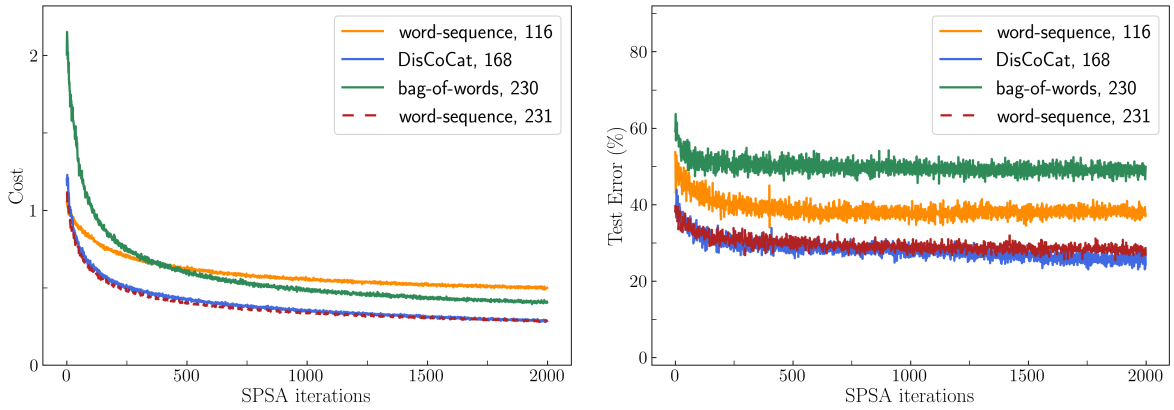
Figure 15 shows the results for the *MC* task, and Figure 16 for the *RP* task for the comparisons outlined above. As can be seen, in the *MC* task, being about the presence of certain words to identify the class the sentence is in, the bag-of-words model does best both with more and with fewer parameters than the DisCoCat model. The word-sequence model does slightly worse than DisCoCat both in the cost function convergence for the training data and for the test errors.

In the *RP* task, however, being more about syntactic structure, the DisCoCat model is on a par with the 231 parameter word-sequence model (and even has a slightly lower test error than the latter), despite having only 168 parameters. Just as expected, the bag-of-words model does not do better than random guessing for the test set evaluations. Also note that the word-sequence model with fewer parameters performs worse than DisCoCat.



(a) (b)

Figure 15: Convergence of various models in the classical simulation (averaged over 20 runs) showing (a) the cost function and (b) the test error for the *MC* task.



(a) (b)

Figure 16: Convergence of various models in the classical simulation (averaged over 20 runs) showing (a) the cost function and (b) the test error for the *RP* task.

7.4 A Sanity Check

In light of how similarly the word-sequence and the DisCoCat model perform on the *RP* task with its dataset suffering from biases due to its small size (relative to the size of the vocabulary), and in order to demonstrate the difference in syntax-sensitivity that the respective models should have by definition, this section presents an additional task on an entirely artificial dataset as a sanity-check of our understanding. The basic idea is to consider the same task as in *RP*, but on a dataset that is designed to prevent the model from potentially just picking up the occurrence of certain words that signify the class, instead of actually learning the sentence order.

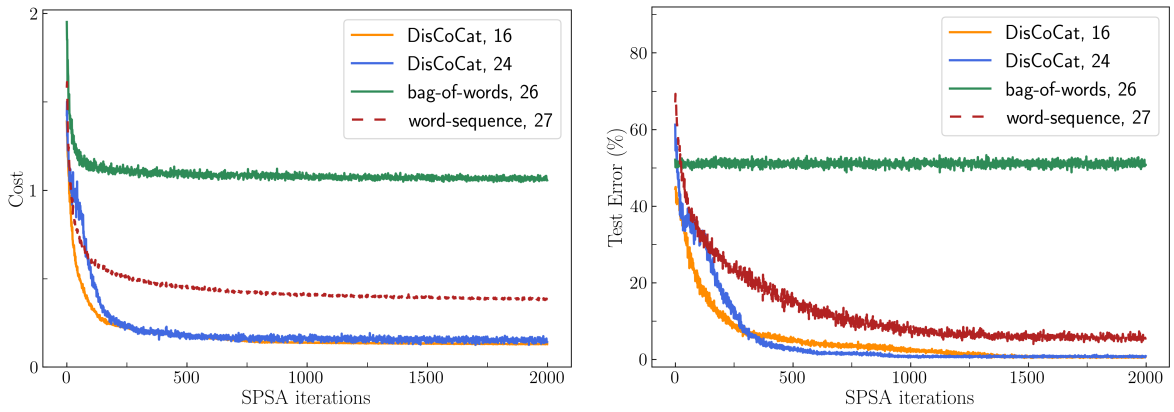
The dataset was constructed on the basis of a vocabulary of 13 words with 8 nouns, 4 transitive verbs and the relative pronoun 'that', to yield a perfectly balanced dataset

in the following way. For any possible choice of a triple $(n1, n2, v)$ of two distinct nouns and one verb, all four combinatorially possible noun phrases are created that are of one of the two syntactic structures that appear in the *RP* dataset. For instance, given the words 'organisation' ($n1$), 'teacher' ($n2$) and 'support' (v) the four phrases are (writing rp for 'that'):

'organisation that support teacher' ($n1$ rp v $n2$),
'teacher that support organisation' ($n2$ rp v $n1$),
'organisation that teacher support' ($n1$ rp $n2$ v),
'teacher that organisation support' ($n2$ rp $n1$ v).

Hence, each noun and each verb appear by construction an equal amount of times in the subject and the object subclause cases (in an overall dataset of size 448 noun phrases). This does of course lead to some sentences which do not make sense such as 'building which like document', but the purpose of this task is not to be a linguistically interesting one – it is only to test the performance of the models once statistical effects can be excluded.

The models we choose to compare are the (0,1,2) DisCoCat model (with 16 parameters), the (0,1,4) and the (0,2,2) bag-of-words models (with 24 and 26 parameters, respectively) and the (0,1,2) word-sequence model (27 parameters). Note that the much smaller parameter numbers are due to the very limited vocabulary in this task, compared to the original *RP* task. The results are shown in Figure 17.

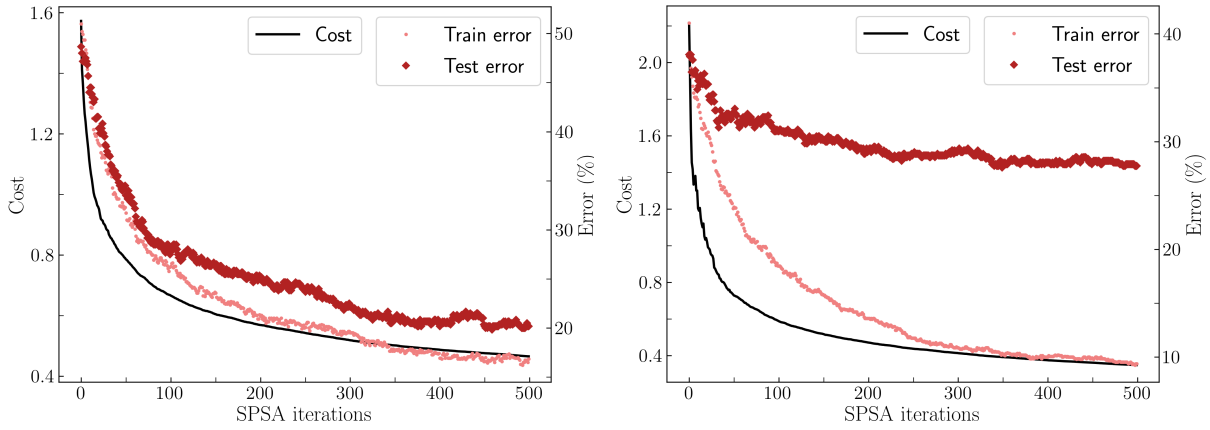


(a) (b)

Figure 17: Convergence of various models in the classical simulation (averaged over 20 runs) showing (a) the cost function and (b) the test error for our sanity-check task with a large and artificial, but perfectly symmetric *RP* like dataset.

Just as expected, the bag-of-words model fails completely. The word-sequence model does converge and learn, but is well out-performed by the DisCoCat model for both chosen ansätze, even though both choices of settings lead to a model with fewer parameters than the word-sequence model.

The performances of our three models on all three tasks, *MC*, *RP* and this sanity-check task here, taken together, confirm the choice of the DisCoCat model as an economical and versatile choice of model for the experiments on actual quantum hardware.



(a) (b)

Figure 18: Classical simulation results for the cost and errors (again averaged over 20 runs) for the DisCoCat model, in (a) for the *MC* task and the chosen ansatz $(1, 3, 1)$ and in (b) for the *RP* task and the chosen ansatz $(0, 1, 2)$.

7.5 Quantum Runs

We now turn to the experiments on actual quantum hardware, which concern the DisCoCat model for the two chosen ansätze for the two respective tasks (see Sec. 7.3). Note that the reason for not running the two baseline models on quantum hardware is that (especially at the time when the experiments were run) access to quantum hardware is costly and limited, in conjunction with the fact that additionally running the baseline models on quantum hardware would not yield any further scientific insight. The question of which models are fit for the chosen tasks can be answered through classical simulations – see above analyses for the conclusion in favour of the DisCoCat model. As for studying how the pipeline performs when actually running on quantum hardware, that is, going from theory to experiment, which often incurs technical challenges and surprises, e.g., due to the noisiness, implementing the DisCoCat model suffices. Ignoring the conceptual differences between the latter and the baseline models, which is completely understood from simulations, the actual quantum circuits – their structure and typical gates – do not differ in any significant way as far as the implementability is concerned due to consistently using the same ansätze throughout.

Just before finally presenting the results of the quantum runs, Figs. 18a and 18b show simulation results for the correspondingly chosen DisCoCat ansätze together with the errors, but for fewer iterations than in Figs. 14a and 14b for better visibility and a more detailed comparison with the quantum runs. After 500 iterations in the *MC* case, the train and test errors read 16.9% and 20.2%, respectively; in the *RP* case the train and test errors are 9.4% and 27.7%, respectively. Noticeably, the test error for the latter task is somewhat higher than the test error for the former task. This is as expected with one of the most important reasons being the large vocabulary in combination with the small size of the

dataset; for example, analysing the data in the aftermath revealed that many of the 115 words in the vocabulary appear only in the test set $\mathcal{P}$, but not at all in the training set $\mathcal{T}$.[26]

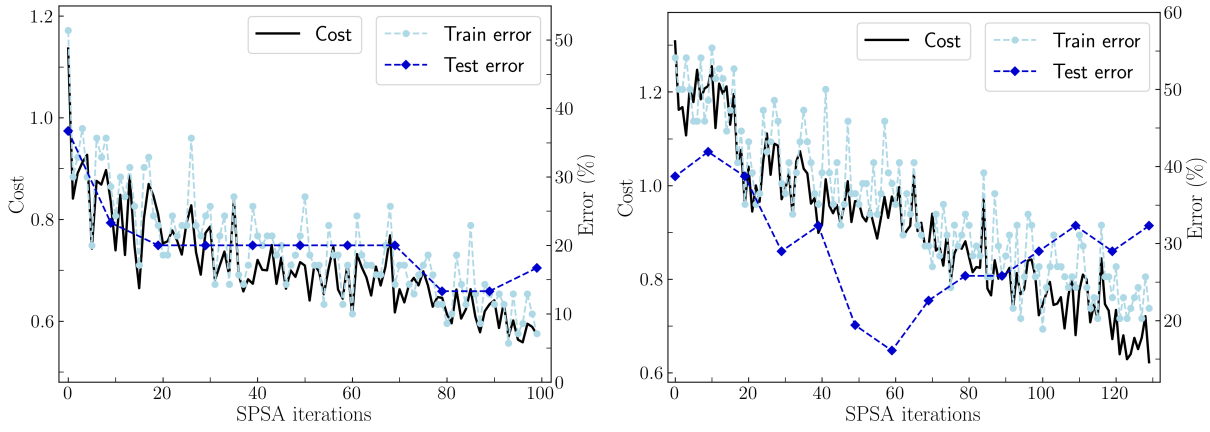
For both tasks executed on quantum hardware, all circuits (compiled with TKET$^{\text{TM}}$) were run on IBM's machine `ibmq_bogota`. This is a superconducting quantum computing device with 5 qubits and quantum volume 32.[27] As explained before, due to the limited access on quantum hardware, we only performed experiments with the DisCoCat model.

Every time the value of the cost or the errors are to be calculated, the compiled circuits corresponding to all sentences or phrases in the corresponding dataset ($\mathcal{T}$, $\mathcal{D}$ or $\mathcal{P}$) are sent as a single job to IBM's device. There, each circuit is run 2^{13} times (the maximum number permitted at the time). The returned data thus comprises for each circuit (involving q qubits) $2^{13} \times q$ measurement outcomes (0 or 1). As explained in detail in Sec. 5 (also see Sec. 4 for the general idea) the pipeline then involves for every sentence or phrase P appropriately post-selecting the data by restricting the measurement data to the 0 outcomes on all those qubits that feature a 0-effect in P's circuit diagram.[28] Once post-selected, the relative frequencies of outcomes 0 and 1 of the remaining qubit that carries the output state of P (corresponding to the wire without a 0-effect in P's circuit diagram), give the estimate of $|\langle i|P(\Theta)\rangle|^2$ (with $i = 0, 1$) and thus of $l_\Theta(P)$. The remaining post-processing to calculate the cost or an error is then as for the classical simulation.

The experiments involved one single run of minimising the cost over 100 iterations for the *MC* task and 130 iterations for the *RP* task, in each case with an initial parameter point that was chosen on the basis of simulated runs on the train (and dev) datasets.[29] For the *MC* task, obtaining all the results shown in Fig. 19a took just under 12 hours of run time. This was enabled by having exclusive access to `ibmq_bogota` for this period of time. In contrast, the *RP* jobs were run in IBMQ's 'fairshare' mode, i.e. with a queuing system in place that ensures fair access to the machine for different researchers. As a consequence, for the *RP* task, which is not computationally more involved than the *MC* task, obtaining all the results shown in Fig. 19b took around 72 hours. With access to quantum devices still being a limited resource and 'exclusive access' being rationed, we see here the reason for the problems that the time cost of yet larger datasets would entail.

Figures 19a and 19b show the cost and various errors for the *MC* task with ansatz $(1, 3, 1)$ and for the *RP* task with ansatz $(0, 1, 2)$, respectively. Despite the noise levels that come with NISQ era quantum computers, and given the fact of a single run with few iterations compared to the classical simulations in Sec. 7.3, the results look remarkably good – the cost is decreasing with SPSA iterations in both cases, modulo the expected fluctuations.

26. More precisely, 17% (36%) of the vocabulary appear zero times (once) in $\mathcal{T}$.
27. Quantum volume is a metric which allows quantum computers of different architectures to be compared in terms of overall performance, and it quantifies the largest random circuit of equal width and depth that a quantum computer can successfully implement.
28. Note that the *MC* dataset includes sentences that lead to circuits on only three qubits. Here 'appropriately post-selecting' means post-selecting two out of the three used qubits.
29. This choice was made to reduce the chances of being particularly unlucky with the one run that we did on actual quantum hardware. Yet, this choice's significance should not be overrated given that the influence of quantum noise spoils the predictability of the cost at a particular parameter point from simulated data.



(a) (b)

Figure 19: Results from quantum computation for the cost, as well as train and test errors (test error for every 10th iteration) for the DisCoCat model, in (a) for the *MC* task and chosen ansatz $(1, 3, 1)$ and in (b) for the *RP* task and chosen ansatz $(0, 1, 2)$.

After 100 (130) iterations as reported in Fig. 19a (Fig. 19b), the test error was 16.7% (32.3%) for the *MC* and *RP* task, respectively, with F-score 0.85 (0.75). These results were checked to be statistically significant against random guessing with $p \leq 0.001$ for *MC* and $p \leq 0.10$ for *RP* according to a permutation test.

Compared to the simulations (Figs. 18a, 18b), it can be seen that after the same number of iterations, test errors are actually lower for the quantum runs. However, due to the special conditions under which these experiments were performed (single run on quantum hardware subject to quantum noise versus many averaged runs on classical hardware without noise, but with the inherent instability of SPSA optimization still present), such comparisons are not very conclusive. In general, the trends presented in the plots of Figure 19 are the expected based on the size of the datasets. For example, the test error in Fig. 19b shows a paradigmatic example of overfitting around iteration 60.

8. Future Work and Conclusions

In this work we have provided a detailed exposition of two experiments of NLP tasks implemented on small noisy quantum computers. Our main goal was to present novel larger-scale experiments building on prior proof-of-concept work (Meichanetzidis et al., 2023), while having in mind the AI/NLP practitioner. We provided for the first time quantum versions of three compositional models, each entailing a different degree of syntax sensitivity, and we tested them successfully on well-defined, although simple in nature, NLP tasks. Despite the prototypical nature of the currently available, albeit rapidly growing in size and quality, quantum processors, we obtain meaningful results on our datasets. We also hope that the current exposition will serve as a useful introduction to practical QNLP for the AI/NLP community.

Having established a QNLP framework for near-term quantum hardware, we briefly outline directions for future work. The ansatz circuits we have used to parameterise the word meanings served well for this work's goal and also are motivated in the QML literature by the fact that they are conjectured to be hard to simulate classically. However, it was beyond the scope of this work to search for optimal word-circuits in a task-specific way. This opens up an exploratory arena for future work on ansätze. In particular, an open question regards trade-offs of performance of ansatz families in a specific task versus general performance on many tasks.

Furthermore, a crucial direction for further work regards scalability and resource estimation. There is of course more than one way that one can think of scaling up QNLP tasks. What is special to QNLP is how scaling up in different dimensions manifests itself as different resource costs in the context of quantum computation, especially given the modest quantum devices available today. First of all, we can consider the cost as the sentences get longer. As a sentence scales in length, the number of qubits on which its corresponding quantum circuit is defined, i.e. the circuit width, will scale as well, depending on the number of qubits assigned to each pregroup type. This consideration is remedied by the realisation that quantum computers have been growing in qubit numbers and there is no sign of this growth slowing down. More importantly, however, a longer sentence will incur an exponential time-cost in the number of qubits being post-selected. Note that in the long term, one does not aim to post-select, but employ more sophisticated protocols where only one qubit needs to be measured, resulting in additive approximations of an amplitude encoding a tensor contraction (Arad & Landau, 2010). Of course, in natural language, sentences are usually upper bounded in length anyway so that we can consider these as up-front constant costs.

Moreover, the DisCoCat framework is applicable to other typological grammars. For example, parsing of a sentence with a CCG grammar returns a syntax tree decorated with grammatical types. For such tree structures one can again define a 'syntactic functor', as done in this work, and instantiate them as quantum circuits for NLP tasks. Experimenting with other grammars than pregroup grammars opens up a playground for defining a whole landscape of QNLP models. Our `lambeq` Python package serves as the platform to begin exploring the possible species of compositional QNLP models. Also note that in our current setup, the number of parameters that one needs to optimise scales polynomially with the size of the vocabulary. This motivates the careful study of the landscapes defined by a task's cost function, as well as the exploration of other optimisation methods beyond SPSA. Another important direction is that we aim to experiment with pre-trained quantum word embeddings, instead of training word-states from scratch for each task, as done in this work. These can then be used in any downstream QNLP task and their training would be considered an upfront cost. We expect that using pre-trained quantum word embeddings would be beneficial, seeing as it has been shown to be beneficial to employ pre-trained word embeddings in classical NLP pipelines.

Hence, all of the above ideas and developments taken together open up the possibility of readily using large-scale textual data in NLP tasks relevant to the real world, moving away from the proof-of-concept setting of this work, whose motivation was only to introduce this novel approach to both the quantum computing and NLP communities.

Before closing, worthy of a comment is the apparent linearity of the model studied in this work. Indeed, quantum theory is a linear theory, at least as far as the unitary evolution of pure states is concerned. However, the subtlety lies in how one chooses to embed the input data, in this case the word meanings, and how these embeddings enter the cost function. In this work the word embeddings are defined as pure quantum states and the labels that enter the cost function are defined in terms of the probability distribution over the measurement outcomes for the particular qubit that carries a sentence's or phrase's representation. Importantly, the Born rule, which gives the probabilities, is a non-linear function of the amplitudes defined in terms of the pure states. More generally, quantum machine learning with variational circuits can be viewed elegantly in terms of kernel methods (Schuld & Killoran, 2019; Schuld, 2021). In this light, it becomes clear that the mapping from the parameters defining the input data to the cost is non-linear.

Relating this then to a potential quantum advantage, a possible avenue for obtaining a quantum advantage arises when a QNLP task is designed so that the evaluation of the cost function is hard to simulate classically due to a high degree of entanglement and interference, all the while demonstrating better performance than the classical state of the art methods. Another avenue to quantum advantage that we aim to investigate is defining compositional models such that any variational optimisation in the pipeline involves circuits that are small enough to be efficiently simulatable classically, but also such that when these components are composed to form a larger quantum-circuit-based model, a quantum computer would be needed to evaluate it and perform the QNLP task at hand. These types of quantum advantage in the field of NLP would be 'meaningful' in that they would constitute examples of non-contrived real-world applications of quantum computers in the near-term.

It is with a view to such advantages that experimenting with NLP models on quantum computing hardware matters now, so that we are prepared for the technological breakthroughs as they happen. There is always a gap between theory and practice and while there is a considerable body of work on the theoretical side of things, practical experimental implementations remain largely unexplored within QNLP. Quantum computers are here to stay and it is thus imperative to start exploring what they can offer the field of NLP and, more generally, AI.

Acknowledgments

We are grateful to Richie Yeung for his help on technical issues, and also, along with Alexis Toumi, for DisCoPy support. We also thank Marcello Benedetti for helpful discussions. We would furthermore like to thank Quantinuum's TKET^{TM} team for support with `PyTKET`. We acknowledge the use of IBM Quantum services for this work. 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

Abramsky, S., & Coecke, B. (2004). A Categorical Semantics of Quantum Protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science*, pp. 415–425. IEEE Computer Science Press. arXiv:quant-ph/0402130.

Arad, I., & Landau, Z. (2010). Quantum Computation and the Evaluation of Tensor Networks. *SIAM Journal on Computing*, *39*(7), 3089–3121.

Bankova, D., Coecke, B., Lewis, M., & Marsden, D. (2018). Graded Hyponymy for Compositional Distributional Semantics. *Journal of Language Modelling*, *6*(2), 225–260.

Basile, I., & Tamburini, F. (2017). Towards Quantum Language Models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1840–1849, Copenhagen, Denmark. Association for Computational Linguistics.

Bausch, J., Subramanian, S., & Piddock, S. (2021). A Quantum Search Decoder for Natural Language Processing. *Quantum Machine Intelligence*, *3*(16), 1–24.

Beer, K., Bondarenko, D., Farrelly, T., Osborne, T. J., Salzmann, R., Scheiermann, D., & Wolf, R. (2020). Training Deep Quantum Neural Networks. *Nature Communications*, *11*(808).

Benedetti, M., Lloyd, E., Sack, S., & Fiorentini, M. (2019). Parameterized Quantum Circuits as Machine Learning Models. *Quantum Science and Technology*, *4*(4), 043001.

Bharti, K., Cervera-Lierta, A., Kyaw, T. H., Haug, T., Alperin-Lea, S., Anand, A., Degroote, M., Heimonen, H., Kottmann, J. S., Menke, T., Mok, W.-K., Sim, S., Kwek, L.-C., & Aspuru-Guzik, A. (2022). Noisy Intermediate-Scale Quantum Algorithms. *Rev. Mod. Phys.*, *94*, 015004.

Brown, B. J., Loss, D., Pachos, J. K., Self, C. N., & Wootton, J. R. (2016). Quantum Memories at Finite Temperature. *Reviews of Modern Physics*, *88*, 045005.

Cao, Y., Romero, J., & Aspuru-Guzik, A. (2018). Potential of Quantum Computing for Drug Discovery. *IBM Journal of Research and Development*, *62*(6), 6:1–6:20.

Cao, Y., Romero, J., Olson, J. P., Degroote, M., Johnson, P. D., Kieferová, M., Kivlichan, I. D., Menke, T., Peropadre, B., Sawaya, N. P. D., Sim, S., Veis, L., & Aspuru-Guzik, A. (2019). Quantum Chemistry in the Age of Quantum Computing. *Chemical Reviews*, *119*(19), 10856–10915.

Chen, Y., Pan, Y., & Dong, D. (2021). Quantum Language Model with Entanglement Embedding for Question Answering. *IEEE Transactions on Cybernetics*, 1–12.

Coecke, B., de Felice, G., Meichanetzidis, K., & Toumi, A. (2020). Foundations for Near-Term Quantum Natural Language Processing. *arXiv preprint arXiv:2012.03755*.

Coecke, B., & Gogioso, S. (2022). *Quantum In Pictures*. Quantinuum.

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

Coecke, B., Sadrzadeh, M., & Clark, S. (2010). Mathematical Foundations for a Compositional Distributional Model of Meaning. *Linguistic Analysis*, *36*, 345–384.

de Felice, G., Toumi, A., & Coecke, B. (2020). DisCoPy: Monoidal Categories in Python. In *Proceedings of the 3rd Annual International Applied Category Theory Conference*. EPTCS.

Ferguson, R. R., Dellantonio, L., Balushi, A. A., Jansen, K., Dür, W., & Muschik, C. A. (2021). Measurement-Based Variational Quantum Eigensolver. *Phys. Rev. Lett.*, *126*, 220501.

Gallego, Á. J., & Orús, R. (2022). Language Design as Information Renormalization. *SN Computer Science*, *3*(140).

Grefenstette, E., & Sadrzadeh, M. (2011). Experimental Support for a Categorical Compositional Distributional Model of Meaning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1394–1404. Association for Computational Linguistics.

Gupta, S., & Zia, R. (2001). Quantum Neural Networks. *Journal of Computer and System Sciences*, *63*(3), 355–383.

Harrow, A. W., & Napp, J. C. (2021). Low-Depth Gradient Measurements Can Improve Convergence in Variational Hybrid Quantum-Classical Algorithms. *Phys. Rev. Lett.*, *126*, 140502.

Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., & Gambetta, J. M. (2019). Supervised Learning with Quantum-Enhanced Feature Spaces. *Nature*, *567*, 209–212.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780.

Huang, H.-Y., Broughton, M., Mohseni, M., Babbush, R., Boixo, S., Neven, H., & McClean, J. R. (2021). Power of Data in Quantum Machine Learning. *Nature Communications*, *12*(2631), 1–9.

Iyyer, M., Manjunatha, V., Boyd-Graber, J., & Daumé III, H. (2015). Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long papers)*, pp. 1681–1691.

Jaderberg, B., Anderson, L. W., Xie, W., Albanie, S., Kiffner, M., & Jaksch, D. (2022). Quantum Self-Supervised Learning. *Quantum Science and Technology*, *7*(3), 035005.

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. *arXiv preprint arXiv:2110.04236*.

Kartsaklis, D., & Sadrzadeh, M. (2014). A Study of Entanglement in a Categorical Framework of Natural Language. In Coecke, B., Hasuo, I., & Panangaden, P. (Eds.), *Quantum Physics and Logic 2014 (QPL 2014). EPTSC 172*, pp. 249–261.

Kartsaklis, D., Sadrzadeh, M., & Pulman, S. (2012). A Unified Sentence Space for Categorical Distributional-Compositional Semantics: Theory and Experiments. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Posters, 8-15 December 2012, Mumbai, India*, pp. 549–558.

Kartsaklis, D., Sadrzadeh, M., Pulman, S., & Coecke, B. (2016). *Reasoning about Meaning in Natural Language with Compact Closed Categories and Frobenius Algebras*, p. 199–222. Lecture Notes in Logic. Cambridge University Press.

Lambek, J. (2008). *From Word to Sentence*. Polimetrica, Milan.

Lewis, M. (2019). Modelling Hyponymy for DisCoCat. In *Proceedings of the Applied Category Theory Conference*, Oxford, UK.

Meichanetzidis, K., Gogioso, S., De Felice, G., Chiappori, N., Toumi, A., & Coecke, B. (2021). Quantum Natural Language Processing on Near-Term Quantum Computers. *Quantum Physics and Logic 2020 (QPL 2020) EPTCS 340*, 213–229.

Meichanetzidis, K., Toumi, A., de Felice, G., & Coecke, B. (2023). Grammar-Aware Sentence Classification on Quantum Computers. *Quantum Machine Intelligence*, *5*(10), 1–16.

Mitchell, J., & Lapata, M. (2010). Composition in Distributional Models of Semantics. *Cognitive Science*, *34*(8), 1388–1429.

Nielsen, M. A., & Chuang, I. L. (2011). *Quantum Computation and Quantum Information: 10th Anniversary Edition* (10th edition). Cambridge University Press, New York, NY, USA.

Orús, R. (2014). A Practical Introduction to Tensor Networks: Matrix Product States and Projected Entangled Pair States. *Annals of Physics*, *349*, 117–158.

O'Riordan, L. J., Doyle, M., Baruffa, F., & Kannan, V. (2020). A Hybrid Classical-Quantum Workflow for Natural Language Processing. *Machine Learning: Science and Technology*, *2*(1), 015011.

Piedeleu, R., Kartsaklis, D., Coecke, B., & Sadrzadeh, M. (2015). Open System Categorical Quantum Semantics in Natural Language Processing. In *Proceedings of the 6th Conference on Algebra and Coalgebra in Computer Science*, Nijmegen, Netherlands.

Pirandola, S., Andersen, U. L., Banchi, L., Berta, M., Bunandar, D., Colbeck, R., Englund, D., Gehring, T., Lupo, C., Ottaviani, C., & et al. (2020). Advances in Quantum Cryptography. *Advances in Optics and Photonics*, *12*(4), 1012–1236.

Preskill, J. (2018). Quantum Computing in the NISQ era and beyond. *Quantum*, *2*, 79.

Ramesh, H., & Vinay, V. (2003). String Matching in O(n+m) Quantum Time. *Journal of Discrete Algorithms*, *1*(1), 103–110. Combinatorial Algorithms.

Rimell, L., Maillard, J., Polajnar, T., & Clark, S. (2016). RELPRON: A Relative Clause Evaluation Data Set for Compositional Distributional Semantics. *Computational Linguistics*, *42*(4), 661–701.

Sadrzadeh, M., Clark, S., & Coecke, B. (2013). The Frobenius Anatomy of Word Meanings I: Subject and Object Relative Pronouns. *Journal of Logic and Computation*, *23*(6), 1293–1317.

Sadrzadeh, M., Clark, S., & Coecke, B. (2014). The Frobenius Anatomy of Word Meanings II: Possessive Relative Pronouns. *Journal of Logic and Computation*, *26*(2), 785–815.

Sadrzadeh, M., Kartsaklis, D., & Balkır, E. (2018). Sentence Entailment in Compositional Distributional Semantics. *Annals of Mathematics and Artificial Intelligence*, *82*, 189–218.

Schuld, M. (2021). Supervised Quantum Machine Learning Models are Kernel Methods. *arXiv preprint arXiv:2101.11020*.

Schuld, M., & Killoran, N. (2019). Quantum Machine Learning in Feature Hilbert Spaces. *Physical Review Letters*, *122*(4), 040504.

Sivarajah, S., Dilkes, S., Cowtan, A., Simmons, W., Edgington, A., & Duncan, R. (2020). t|ket>: a Retargetable Compiler for NISQ Devices. *Quantum Science and Technology*, *6*(1), 014003.

Socher, R., Huval, B., Manning, C., & Ng, A. (2012). Semantic Compositionality through Recursive Matrix-Vector Spaces. In *Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning 2012*, pp. 1201–1211.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., 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*, pp. 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Spall, J. C. (1998). Implementation of the Simultaneous Perturbation Algorithm for Stochastic Optimization. *IEEE Transactions on Aerospace and Electronic Systems*, *34*(3), 817–823.

Steedman, M. (2001). *The Syntactic Process*. MIT Press.

Wiebe, N., Bocharov, A., Smolensky, P., Troyer, M., & Svore, K. M. (2019). Quantum Language Processing. *arXiv preprint arXiv:1902.05162*.

Yeung, R., & Kartsaklis, D. (2021). A CCG-Based Version of the DisCoCat Framework. In *Proceedings of the 2021 Workshop on Semantic Spaces at the Intersection of NLP, Physics, and Cognitive Science (SemSpace)*, pp. 20–31, Groningen, The Netherlands. Association for Computational Linguistics.

Zeng, W., & Coecke, B. (2016). Quantum Algorithms for Compositional Natural Language Processing. *Electronic Proceedings in Theoretical Computer Science*, *221*, 67–75.