Quantum Natural Language Generation on Near-Term Devices

Amin Karamlou[*]
IBM Quantum
University of Oxford

Marcel Pfaffhauser
IBM Quantum

James Wootton
IBM Quantum

Abstract

The emergence of noisy medium-scale quantum devices has led to proof-of-concept applications for quantum computing in various domains. Examples include Natural Language Processing (NLP) where sentence classification experiments have been carried out, as well as procedural generation, where tasks such as geopolitical map creation, and image manipulation have been performed. We explore applications at the intersection of these two areas by designing a hybrid quantum-classical algorithm for sentence generation.

Our algorithm is based on the well-known simulated annealing technique for combinatorial optimisation. An implementation is provided and used to demonstrate successful sentence generation on both simulated and real quantum hardware. A variant of our algorithm can also be used for music generation.

This paper aims to be self-contained, introducing all the necessary background on NLP and quantum computing along the way.

1 Introduction

It is widely believed that computers operating according to the laws of quantum mechanics will outperform classical computers at specialised tasks. This belief is backed up by the fact that important computational problems such as integer factorisation (Shor, 1997) and unstructured search (Grover, 1996) admit quantum algorithms which are provably faster than the best known classical algorithms for solving them. Unfortunately, in order to make use of these algorithms, we would first need to build scalable, fault-tolerant quantum computers, which are still some years away. By contrast, the current generation of quantum computers are still fairly rudimentary, containing at most a few hundred noisy qubits, i.e. qubits with which we cannot

_____
[*]corresponding author: Amin.Karamlou@cs.ox.ac.uk.

perform perfect operations (Preskill, 2018). Despite their shortcomings, these devices represent a significant milestone for quantum computing. This is because unlike their smaller predecessors they cannot be simulated efficiently on classical hardware. Hence, it is possible that near-term quantum devices will bring with them the first examples of tasks performed by quantum computers that not even the most powerful classical supercomputers can perform, with tentative first steps made for proof-of-principle problems (Arute et al., 2019; Pednault et al., 2019). The search for examples in which a useful advantage can be demonstrated has led to the development of tailor-made algorithms for near-term devices that solve problems in domains such as chemistry, and optimisation (Farhi et al., 2014; Peruzzo et al., 2014).

In this paper, we are concerned with near-term quantum algorithms for natural language generation (NLG). NLG lies at the intersection of procedural generation, i.e. the algorithmic generation of data, and Natural Language Processing (NLP), both of which are active research topics within the quantum software community (see e.g. Wootton, 2020b,a; Coecke et al., 2020; Lorenz et al., 2021). The importance of NLG is underscored by its wide range of potential applications. It can for instance be used in video games to create natural-sounding dialogue, or in journalism to create automated news articles. These applications are often time-sensitive, as in the case of video games, where delays in dialogue generation would make the user experience unsatisfactory. In other situations, NLG algorithms have to deal with a large amount of input data. This is the case in automated journalism where information from many different sources needs to be collated into one coherent article. These considerations mean that developing faster algorithms for NLG tasks would have tremendous practical consequences. Thus, it is natural to wonder if any such tasks can benefit from speedups when performed

on a quantum computer. Our aim here is to take the first steps towards answering this question.

Throughout this work, we will make use of the well-established mathematical connection between the Distributional Compositional Categorical (DisCoCat) (Coecke et al., 2010) model of natural language and quantum theory. This connection was recently exploited in several works (Meichanetzidis et al., 2020; Lorenz et al., 2021) to successfully perform Quantum Natural Language Processing (QNLP) on real quantum hardware (as opposed to simulation with conventional hardware). More specifically it was used to perform the task of binary sentence classification. The aim of this task is simple: Given a sentence about one of two possible topics, decide which topic it is about. Building upon this work, we design a sentence generation algorithm that can run on current quantum hardware. Our algorithm takes as input one of several possible topics and produces as output a sentence with that topic. Our algorithm works by searching through the space of possible sentences using simulated annealing (SA), a well-known probabilistic method for solving combinatorial optimisation problems. The choice of SA is motivated by the recent success of the method at (classically) solving the task of sentence paraphrasing (Liu et al., 2020). We experimentally evaluate the performance of our algorithm at news headline generation. We also show how our algorithm can be adapted to perform music generation.

Before continuing it is worth clarifying the goal of this paper and the scope of our claims. The formal similarity between DisCoCat and quantum theory has led to some authors claiming that NLP is an inherently "quantum native" field (Coecke et al., 2020), and that we can expect large-scale quantum computational speedups for NLP tasks as more powerful quantum hardware becomes available. Testing these claims theoretically would require significant analysis of QNLP proposals using computational complexity theory, as has been done with other proposals for quantum advantage, for example in Aaronson and Chen (2016); Brakerski et al. (2020); Zhu et al. (2021). Alternatively, we could wait for larger quantum computers to be built, allowing for experimental comparison of QNLP algorithms and cutting-edge classical methods such as GPT-3 (Brown et al., 2020) or BERT (Devlin et al., 2019). We do not claim to address either one of these challenges here. Our work is rather a proof-of-concept example of how NLG can be performed on quantum hardware. We also hope that by assuming a modest mathematical background this paper can serve as an introduction to quantum software design using the diagrammatic style of quantum theory utilised in QNLP research.

The rest of the paper is organised as follows: In section 2 we describe the necessary background on DisCoCat and quantum computing. Section 3 contains the details of our SA-based sentence generation algorithm. We report the results of experiments with this algorithm in section 4, including a discussion of how the algorithm can be adapted for music composition in section 3.3. Finally, we discuss future research avenues in section 5.

2 Preliminaries

2.1 Quantum Computing

This section presents a self-contained overview of the basics of quantum computation, assuming no familiarity with the topic. Naturally, what we present is far from a complete introduction. A more in-depth book for further reading is Nielsen and Chuang (2002). Alternatively, Coecke and Kissinger (2018) introduces quantum theory via the diagrammatic language used here.

The idea behind quantum computation is to harness features of quantum mechanics that have no classical analogue in the design of efficient algorithms. The first of these features worth mentioning is called *superposition*. The logical building blocks of a classical computer are bits. These are objects that can have one of two possible states, 0 or 1. The quantum analogue of a bit, known as a qubit, has a state that lives in a 2-dimensional Hilbert space. We use the notation[1] $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ to denote the orthonormal basis vectors of this space. The state of a qubit, written as $|\psi\rangle$, is a linear combination of these basis vectors:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \text{ s.t. } \alpha, \beta \in \mathbb{C}, |\alpha|^2 + |\beta|^2 = 1$$

It is this linear combination that is referred to as a superposition.

The act of reading the value of a qubit in state $|\psi\rangle$ is called a *measurement*. Regardless of what

---

[1] This is referred to as Dirac or bra-ket notation and is used ubiquitously throughout quantum information. See appendix 10 of (De Wolf, 2019) for a concise introduction to this formalism.

superposition a qubit is in, the result of a measurement is always one of two possible outcomes, 0 or 1. The probability of measuring 0 is equal to $|\alpha|^2$, and α is known as the *amplitude* of $|0\rangle$. Likewise, the probability of measuring 1 is $|\beta|^2$, and β is known as the *amplitude* of $|1\rangle$. Crucially, once a measurement has occured, the state $|\psi\rangle$ *collapses* to the corresponding basis state. For example, if we measure a qubit in state $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and observe the outcome 0, then immediately after the measurement the state of the qubit is $|0\rangle$.

Naturally, to perform a meaningful computation we need to use more than just one qubit. The *joint state* $|\phi\rangle$ of n qubits lives in a Hilbert space of dimension $N = 2^n$ with orthogonal basis states of the form $|b_1\rangle \otimes |b_2\rangle \otimes ... \otimes |b_n\rangle$ where each $b_i \in \{0, 1\}$. We will abbreviate these basis states to $|b_1 b_2 b_3 ... b_n\rangle$. With some abuse of notation it will also often be convenient to write these basis states in decimal notation i.e. $|0\rangle = |000...000\rangle$, $|1\rangle = |000...001\rangle$, $|2\rangle = |000...010\rangle$, ... $|N-1\rangle = |111...111\rangle$.

$|\phi\rangle$ is then once again a superposition:

$$|\phi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle + ...\alpha_{N-1} |N-1\rangle$$
$$\text{s.t. } \forall i \alpha_i, \beta \in \mathbb{C}, \sum_i |\alpha_i|^2 = 1$$

When measuring $|\phi\rangle$ one observes outcome i with probability $|\alpha_i|^2$ and the state of the underlying qubits collapses to $|i\rangle$.

Aside from measurement, a quantum system can also be manipulated using *quantum logic gates*. Mathematically, these gates are unitary linear maps U. Thus, the evolution of a system from one timestamp to the next can simply be described as $|\psi_1\rangle = U |\psi_0\rangle$.

Pictorially, a quantum computation can be represented as a circuit. Figure 1 provides an example of such a circuit. In this example, two qubits begin in the joint state $|\psi_0\rangle = |0\rangle$. A quantum logic gate $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, known as a hadamard gate is applied to each qubit, transforming the state into $|\psi_1\rangle = H \otimes H |0\rangle = \frac{1}{2}|0\rangle + \frac{1}{2}|1\rangle + \frac{1}{2}|2\rangle + \frac{1}{2}|3\rangle$. Finally, the state is measured, resulting in one of the four possible outputs $0, 1, 2$, or 3 being observed, each with a probability of $\frac{1}{4}$. After measurement, the state collapses to the respective basis state $|0\rangle, |1\rangle, |2\rangle$, or $|3\rangle$.
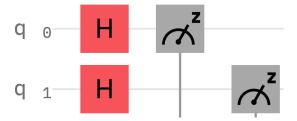

Figure 1: A simple quantum circuit created using the IBM Quantum Composer available at `https://quantum-computing.ibm.com/`.

2.2 DisCoCat and QNLP

The Distributional Compositional Categorical (DisCoCat) model of language meaning (Coecke et al., 2010) is a mathematical framework that allows for the meaning of a sentence to be described as a combination of the meaning of its constituent words, and the grammatical relationships between these words. This is in contrast to many older NLP models, which treat sentences as "bags of words" while ignoring their grammatical structure.

DisCoCat comes equipped with a pictorial representation, allowing any sentence to be represented by a so-called *string diagram*. Such a diagram consists of boxes representing words, and wires connecting these boxes according to the formalism of pregroup grammars (Lambek, 2008). This means that every wire in the diagram is annotated either by some atomic type p, a left adjoint $p.l$, or a right adjoint $p.r$. Let us explain the role of types and adjoints through example, by considering the sentence "Alice generates language". The DisCoCat diagram corresponding to this sentence is given in figure 2. In this diagram, wires are annotated by the noun type n and the sentence type s. As we can see, the box for the word 'generates' has three wires coming out of it, which are annotated by $n.r, s$, and $n.l$ respectively. This indicates that the word 'generates' expects to receive a noun on its left (in this case 'Alice'), as well as another noun on its right (in this case 'language') in order to output a grammatical sentence. In general, a sentence is grammatical if its DisCoCat diagram has a single open output wire of type s, as in the example of figure 2.

It is worth noting that DisCoCat diagrams are more than simple pictures. They are based on the rigorous formalism of monoidal categories (Heunen and Vicary, 2019, Chapter 1), which means they are equipped with a diagrammatic calculus. This calculus can be used to rewrite complicated
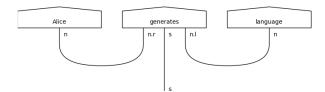
Figure 2: DisCoCat diagram for the sentence 'Alice generates language.'

string diagrams into simpler ones that still encode the meaning of the original sentence. As it happens, monoidal categories and string diagrams also turn out to be a suitable high-level framework for capturing much of quantum information and computation (Abramsky and Coecke, 2004; Coecke and Kissinger, 2018). This observation is part of the reason that one may hope for quantum advantage in NLP tasks in the long term.

We now outline a procedure for transforming any sentence into a parameterised quantum circuit that can be run on real IBM Quantum hardware. The pipeline we discuss here has recently been implemented as part of *lambeq* (Kartsaklis et al., 2021), a python library developed specifically for QNLP tasks.

1. A sentence is converted to a DisCoCat diagram using the Combinatory Categorical Grammar (CCG) based techniques of Yeung and Kartsaklis (2021).

2. The DisCoCat diagram is simplified using some of the rewrite rules available in lambeq. Even though this step is strictly speaking optional, applying rewrite rules often leads to crucial computational advantages, for instance by reducing the number of qubits required to implement the parameterised quantum circuit.

3. An *ansatz* is used to transform the simplified diagram to a parameterised quantum circuit. This ansatz is a mapping that assigns a number of qubits to each wire type in the string diagram, as well as a set of quantum logic gates to each word in the diagram.

4. The quantum compiler t|ket⟩ (Sivarajah et al., 2020) is used to translate the parameterised quantum circuit into machine-specific instructions, which can be executed on real IBM quantum computers.

In this paper we use the IQP ansatz. This transforms each DisCoCat diagram into an Insantanoues

Quantum Polynomial (IQP) circuit. We do not justify this choice of ansatz here, more information is available in (Havlíček et al., 2019; Lorenz et al., 2021). The parameterised quantum circuit corresponding to "Alice generates language" is given in figure 3.

2.3 Sentence Classification

Before we can present our sentence generation algorithm we must first explain how sentence classification can be performed on near-term quantum devices. What we outline here is a step-by-step overview for solving the following task: Given a dataset Γ of sentences, each of which belongs to one of k possible topics, train a classifier that can correctly determine the topic of further unseen sentences (provided the unseen sentences are also about one of the k possible topics). This section mostly follows Lorenz et al. (2021), although we modify the algorithm to perform multi-class rather than binary sentence classification.

1. Each sentence $S \in \Gamma$ is converted to a parameterised quantum circuit C_S using the techniques discussed in the previous section. Note that some parameters may be shared between quantum circuits corresponding to different sentences. This occurs when the same words appear in multiple sentences. We set $q_n = 1$, and $q_s = \lceil \log k \rceil$, where q_n and q_s are the number of qubits associated to the noun and sentence wire types respectively. Measuring such a circuit yields one of k possible outcomes, each of which we associated with one of the topics in our corpus.

2. For each sentence $S \in \Gamma$ and each topic $i \in \{0, 1, ..., k-1\}$ we define a binary predicate $L(i, S) \in \{0, 1\}$ and set $L(i, S) = 1$ if and only if sentence S has topic i. Moreover, we write $P(i, C_S)$ for the probability of observing outcome i when measuring the final state of a quantum circuit C_S. Finally, let Ω denote the full set of parameters used in all the quantum circuits combined. Our goal is thus to find the optimal Ω which maximises $P(i, C_S)$ whenever $L(i, S) = 1$. This problem can be solved using classical machine learning techniques, by minimising the categorical cross-entropy loss function below. This is achieved by using the Simultaneous perturbation stochastic approximation (SPSA) algorithm (Spall, 1998).
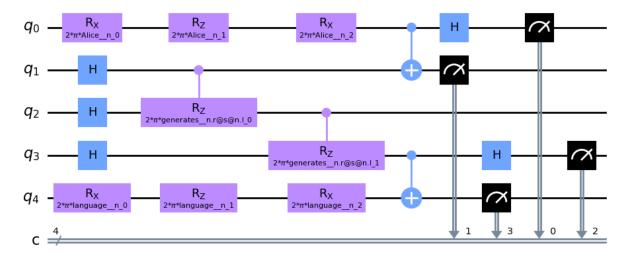
Figure 3: Parameterised quantum circuit for the sentence "Alice generates language".

$$C(\Omega) = \Sigma_{S \in \Gamma} L(i, S). \log P(i, C_S)$$

3. Given an unseen sentence $S \notin \Gamma$ we can now predict its topic as follows: Use the optimal parameters Ω to create the quantum circuit C_S. Measure the final state of C_S obtaining an outcome $i \in \{0, 1, ..., k - 1\}$. Output the topic associated with outcome i.

3 Sentence Generation

In this section, we present our hybrid quantum-classical sentence generation algorithm.

We first discuss the simulated annealing (SA) algorithm for solving combinatorial optimisation problems (Kirkpatrick et al., 1983). Then, we rigorously formulate our sentence generation task as an optimisation problem and show in detail how a version of SA can be used to efficiently generate and test many candidate sentences until a satisfactory one is found.

3.1 Simulated Annealing

An optimisation problem is a problem where a satisfactory solution must be found from a search space of possible solutions. By a satisfactory solution we mean one that maximises (or comes close to maximising) some objective function over the search space.

Simulated annealing (SA) is a well-known heuristic method for solving optimisation problems. Let $\mathcal{X}$ be a search space, and $f : \mathcal{X} \rightarrow [0, 1]$ be an objective function over that search space. The goal of SA is to find $x \in \mathcal{X}$ which maximises

$f(x)$. SA starts by either randomly or heuristically choosing a starting candidate state $x_0 \in \mathcal{X}$. At each step t, the algorithm then considers some neighbouring state x^* of the current candidate x_t. If $f(x^*) > f(x_t)$ then the algorithm 'accepts' x^* by setting $x_{t+1} = x^*$ and beginning a new iteration. In the event that x^* is not accepted SA simply sets $x_{t+1} = x_t$ and begins a new iteration. Even if $f(x^*) <= f(x_t)$ SA may still accept x^* with some small probability $e^{\frac{f(x^*) - f(x_t)}{T}}$. This is known as the *metropolis* criterion and depends on an annealing temperature T. There are many different options available for calculating T at each timestep. Usually this value is set to be high at the start of SA so that x^* has a high acceptance probability. With each iteration, the value of T decreases, allowing SA to converge towards a solution. In this work, we use the *fast simulated annealing* algorithm which sets $T = \frac{T_i}{t+1}$ at each iteration, where T_i is the initial temperature.

Simulated annealing performs well in practice and is guaranteed to converge towards the optimal solution under reasonable assumptions (Granville et al., 1994). Although in the worst-case this convergence may take a prohibitively long amount of time.

3.2 The Algorithm

Let us assume that we have trained a multi-class sentence classifier using the techniques discussed in section 2.3. The sentence generation task we aim to solve is the following: Given as input one of the topics $i \in \{0, 1, ..., k\}$ which the classifier is trained over, produce a sentence with that topic.

This task can be seen as an optimisation problem where the search space $\mathcal{X}$ consists of all sentences formed from the vocabulary used to train the classifier[2]. The objective function f can then simply be defined as $f(S) = P(i, C_S)$. Where C_S is the quantum circuit generated using the optimal parameters Ω. As per the discussion in section 2.3 This function is maximal whenever the sentence S has a high probability of being classified with topic i. We now outline the procedure for solving this optimisation problem using SA.

1. Start by generating a random candidate sentence s_0 from our vocabulary.

2. At each step t we generate a neighbouring state s^* of s_t. This generation proceeds similarly to the word level editing approach of Miao et al. (2019). More specifically, let $s_t = [w_1, w_2, ..., w_n]$. s^* is generated by randomly performing one of the following editing operations:

   - *Insert*: randomly selects a word w and an index j and sets $s^* = [w_1, ...w_{j-1}, w, w_j, ..., w_n]$.
   - *Delete*: randomly selects an index j and sets $s^* = [w_1, ...w_{j-1}, w_{j+1}, ..., w_n]$.
   - *Replace*: randomly selects a word w and an index j and sets $s^* = [w_1, ...w_{j-1}, w, w_{j+1}, ..., w_n]$.

3. Calculate the values $f(s^*) = P(i, C_{s^*})$ and $f(s_t) = P(i, C_{s_t})$ by running the corresponding quantum circuits many times, and building a probability distribution out of the observed outputs. Decide whether to accept s^* or not according to the SA algorithm.

4. Continue iterating until you find a sentence s that passes a high threshold τ along the objective function i.e. $f(s) > \tau$. This indicates that the sentence is with high probability about the topic i as required.

3.3 Application to Music Composition

Much like how a sentence is composed of words placed side by side, a musical composition can be seen as a sequence of music snippets placed next to each other. Each snippet itself is in turn composed of musical notes, similarly to how a word is composed of letters belonging to an alphabet.

This similarity was recently exploited in (Miranda et al., 2021) and used to define a musical version of the DisCoCat framework. The authors then used a CFG to generate a dataset of 100 musical compositions for piano. The generated pieces were annotated manually and placed into one of two classes: rhythmic or melodic. This allowed them to train a quantum classifier that distinguishes rhythmic and melodic musical compositions using the techniques of section 2.3.

By replacing the sentence classifier mentioned in section 3.2 with the musical classifier described above, we can adapt our SA-based algorithm for the task of generating musical compositions. In the future we will make musical compositions created using this technique available on our project Github repository[3].

4 Experiments

We now define and attempt to solve two simple sentence generation tasks using the algorithm from the previous section. Our source code is available at `https://bit.ly/QuantumNLG`. To the best of our knowledge, the only other algorithm that can solve these tasks using a quantum computer is what we shall refer to as the Random Generation and Testing (RGT) method of Miranda et al. (2021). In fact, this algorithm was initially proposed for music composition rather than sentence generation, but it can straightforwardly be adapted to perform the latter task as well. It works by randomly putting words from a vocabulary next to each other, and evaluating the resulting sentence against the objective function we defined in section 3, until a satisfactory sentence is found. We will implement sentence generation using RGT and compare its performance with our SA-based algorithm.

We do not perform any comparison with state-of-the-art classical methods for solving NLG tasks since it is clear that such methods could easily outperform our proof-of-concept algorithm.

4.1 Food vs IT

For our first task, we use the food vs IT data-set created in Lorenz et al. (2021). This dataset consists of 130 sentences generated using a simple Context-Free Grammar (CFG). Each sentence is manually

---

[2]We could even consider the infinite search space of all possible sentences. However, current limitations in quantum hardware mean that solving this more complicated version of the problem is out of scope for the foreseeable future.

[3]`https://github.com/AminKaramlou/QNLG`

labelled as being about one of two possible topics, Food or IT. In Lorenz et al. (2021) a quantum classifier is trained using this dataset according to the techniques discussed in section 2.3. With the help of this classifier, we can implement and analyse the SA and RGT-based sentence generation algorithms on the Food vs IT dataset.

4.1.1 Simulation results

Before performing experiments on real quantum hardware we first run our algorithms on a 'classical simulator'. As the name suggests, this is a classical device that simulates the behaviour of a real quantum computer. Of course, it is prohibitively expensive to simulate large quantum systems (otherwise there would be no point in building quantum devices). Fortunately, the quantum circuits we are dealing with in this paper are all very small, and can thus be simulated efficiently. All simulations in this section were performed on a 2019 MacBook Air with 16 GB of memory and a 1.6 GHz Dual-Core Intel Core i5 processor.

As is standard within NLG literature (Sai et al., 2020) we evaluate the quality of free-form generated sentences using the following two criteria:

1. Correctness: Does the generated sentence have the correct topic?

2. Fluency: Is the generated sentence grammatically and semantically correct?

Table 1 shows the result of using a classical simulator to generate 30 sentences about food. The correctness and fluency of each of these sentences have been determined according to the human judgement of the authors. For instance, the sentence "man debugs software" was judged as being fluent but incorrect while the sentence "tasty person prepares dinner" was judged as being correct but not fluent.

	RGT	SA
Fluent and Correct	23	22
Fluent and Not Correct	0	0
Not Fluent and Correct	4	4
Not Fluent and Not Correct	3	4
Avg No. of guesses	7.56	7.46

Table 1: Results of using a classical simulator to generate 30 sentences about food (Number of guesses refers to the number of candidate sentences evaluated against the objective function by each algorithm).

We can see that both the RGT and SA algorithms have performed similarly in terms of the quality of the produced sentences. This is to be expected given that the acceptance condition for a candidate sentence ($f(s) > \tau$) is the same in both cases. We can also see that the average number of sentences guessed before a valid solution is found is almost the same for both algorithms. This is somewhat surprising, given the more rudimentary nature of RGT compared to SA. We believe the reason for this is the small search space associated with this generation task, as well as the fact that many sentences in this space are actually about food. Thus, RGT has a high likelihood of finding a good sentence in only a few guesses. On the other hand, a poor initial guess in the SA algorithm can be very detrimental in this case, since the algorithm might get stuck in a sub-optimal neighbourhood for a few steps. As we shall see in the news headline generation task, this advantage of RGT quickly disappears when dealing with more complicated search spaces.

4.1.2 Quantum hardware results

We now repeat the experiment above on a real quantum computer, namely IBM's 16 qubit `ibmq_guadalupe` device. When performing experiments on real quantum hardware, it is important to remember that measuring the final state of a quantum circuit will cause this state to collapse to one of the basis states. This means that the only way we can calculate the probabilities $P(i, C_s)$ needed in step 3 of our generation algorithm is to run and measure the circuit C_s repeatedly and create a probability distribution of the observed outcomes. The total number of times a quantum circuit is run in this way is referred to as the number of *shots*. In our case, we ran each circuit for 100000 shots. In the ideal case, results from real quantum hardware will be equivalent to those of simulations. However, imperfections in current prototype devices will lead to sub-optimal performance. The results can therefore be used to benchmark the capacity of current devices for applications of this type.

Table 2 shows the results of using both the RGT and SA algorithms on real quantum hardware in order to generate 10 sentences about food. Interestingly, these results are very similar to the ones obtained using classical simulators in the previous section. This suggests that our algorithms are potentially robust against the inherent noisiness and imperfections of the current generation of quantum computers. We will aim to test this hypothesis fur-

ther with more extensive future experimentation.

	RGT	SA
Fluent and Correct	7	7
Fluent and Not Correct	0	0
Not Fluent and Correct	2	1
Not Fluent and Not Correct	1	2
Avg No. of guesses	8.4	8.5

Table 2: Results of using the 16 qubit `ibmq_guadalupe` quantum computer to generate 10 sentences about food.

4.2 News Headlines

As we have seen both the SA and RGT-based sentence generation algorithms performed fairly well on the Food vs IT dataset. In this section, we will test the behaviour of these algorithms on a more challenging dataset consisting of 105 news headlines. Similarly to (Lorenz et al., 2021), we generated this dataset by using a CFG. The sentences were then manually annotated as belonging to one of four possible news headline topics, *entertainment*, *politics*, *sports*, or *technology*. Compared to the Food vs IT dataset this dataset contains more sentence topics, has a larger vocabulary, and has more complicated CFG production rules. When it comes to sentence generation, this means that there is a much larger search space to consider and that there are fewer acceptable sentences in this search space, making the task significantly more challenging.

Table 3 shows the results of using SA and RGT to generate 30 sentences about politics. As expected for this more complex dataset, the average number of guesses before finding a viable candidate is much less when using SA rather than RGT[4].

	RGT	SA
Timeouts	8	0
Fluent and Correct	1	11
Fluent and Not Correct	4	1
Not Fluent and Correct	3	5
Not Fluent and Not Correct	14	13
Avg No. of guesses	201.1	40.4

Table 3: Results of using a classical simulator to generate 30 sentences about politics (Timeout refers to runs of the algorithm that failed to find a suitable sentence after 500 guesses)

---

[4]Note that we treat timeouts as 500 guesses for the purposes of averaging.

5 Related and Future Work

We have presented a proof-of-concept algorithm showing how a simple NLG task can be performed on current quantum devices. The algorithm also works for generating musical compositions. Two pieces of related work are worth pointing out:

- In Abbaszade et al. (2021) a hybrid quantum-classical algorithm based on DisCoCat is described for sentence translation, a task which has a language generation component to it. Even though the authors do not provide an implementation, this algorithm is well-suited for experimentation on current quantum hardware, as it relies on Quantum Long Short Term Memory (Q-LSTM) (Chen et al., 2020), a quantum machine learning model that is particularly well-suited for near term devices, due to having a modest requirement on qubit counts and circuit depth.

- Arya et al. (2022) formulates the task of music composition as a Quadratic Unconstrained Binary Optimisation (QUBO) problem. QUBO problems are particularly well-suited for being solved using adiabatic quantum computation (AQC) (Farhi et al., 2000). This is an alternative to the circuit-based model we learnt about in section 2[5]. (Arya et al., 2022) then proceeds to solve this QUBO problem using D-Wave quantum computers and generate musical compositions. In future work, it would be interesting to compare this approach to the RGT and SA algorithms we have discussed here.

We conclude with some thoughts on future research directions.

Clearly, all the works above are limited by the small size of today's quantum computers. However, several companies have announced plans for building significantly more powerful quantum devices in the next few years (see e.g. qua, 2020). These devices will undoubtedly be capable of solving more sophisticated NLG tasks than the ones presented here. Whether or not this will eventually lead to quantum algorithms that outperform today's state-of-the-art classical NLG techniques is a fascinating open question that could have dramatic consequences for the field as a whole. We hope

---

[5]Although both models are equivalent in terms of computational power (Aharonov et al., 2008).

that this work serves as sufficient inspiration for the rest of the community to join us in tackling this question.

A further limitation of our techniques is the fact that DisCoCat, while well-suited for modelling the meaning of sentences, is not capable of modelling the meaning of larger pieces of text. This is problematic when it comes to performing more sophisticated NLG tasks e.g. text summarization, given that these tasks often require the production or manipulation of long passages of text. To alleviate this issue, we could use a recently proposed generalisation of DisCoCat, referred to as the Distributional Compositional Circuit-based (DisCoCirc) model (Coecke, 2021). Inspired by how DisCoCat uses the grammatical relationship between words to encode the meaning of a sentence, DisCoCirc uses the relationship between sentences to encode the meaning of an entire passage of text. A potential avenue for future work is thus to use DisCoCirc and create a pipeline similar to what we have seen in sections 2.3 and 3 for solving document-level rather than sentence-level NLG tasks.

References

2020. Ibm's roadmap for scaling quantum technology. https://research.ibm.com/blog/ibm-quantum-roadmap. Accessed: 2022-03-15.

Scott Aaronson and Lijie Chen. 2016. Complexity-theoretic foundations of quantum supremacy experiments. *arXiv preprint arXiv:1612.05903*.

Mina Abbaszade, Vahid Salari, Seyed Shahin Mousavi, Mariam Zomorodi, and Xujuan Zhou. 2021. Application of quantum natural language processing for language translation. *IEEE Access*, 9:130434–130448.

Samson Abramsky and Bob Coecke. 2004. A categorical semantics of quantum protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004.*, pages 415–425. IEEE.

Dorit Aharonov, Wim Van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. 2008. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM review*, 50(4):755–787.

Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. 2019. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510.

Ashish Arya, Ludmila Botelho, Fabiola Cañete, Dhruvi Kapadia, and Özlem Salehi. 2022. Music composition using quantum annealing. *arXiv preprint arXiv:2201.10557*.

Zvika Brakerski, Venkata Koppula, Umesh Vazirani, and Thomas Vidick. 2020. Simpler proofs of quantumness. *arXiv preprint arXiv:2005.04826*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Samuel Yen-Chi Chen, Shinjae Yoo, and Yao-Lung L Fang. 2020. Quantum long short-term memory. *arXiv preprint arXiv:2009.01783*.

Bob Coecke. 2021. The mathematics of text structure. In *Joachim Lambek: The Interplay of Mathematics, Logic, and Linguistics*, pages 181–217. Springer.

Bob Coecke, Giovanni de Felice, Konstantinos Meichanetzidis, and Alexis Toumi. 2020. Foundations for near-term quantum natural language processing.

Bob Coecke and Aleks Kissinger. 2018. Picturing quantum processes. In *International Conference on Theory and Application of Diagrams*, pages 28–31. Springer.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *ArXiv*, abs/1003.4394.

Ronald De Wolf. 2019. Quantum computing: Lecture notes. *arXiv preprint arXiv:1907.09415*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*.

Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. 2000. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*.

V. Granville, M. Krivanek, and J.-P. Rasson. 1994. Simulated annealing: a proof of convergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):652–656.

Lov K. Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA. Association for Computing Machinery.

Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. 2019. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212.

Chris Heunen and Jamie Vicary. 2019. *Categories for Quantum Theory: an introduction*. Oxford University Press.

Dimitri Kartsaklis, Ian Fan, Richie Yeung, Anna Pearson, Robin Lorenz, Alexis Toumi, Giovanni de Felice, Konstantinos Meichanetzidis, Stephen Clark, and Bob Coecke. 2021. lambeq: An efficient high-level python library for quantum nlp.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by simulated annealing. *Science*, 220(4598):671–680.

Joachim Lambek. 2008. *From Word to Sentence: a computational algebraic approach to grammar*. Polimetrica sas.

Xianggen Liu, Lili Mou, Fandong Meng, Hao Zhou, Jie Zhou, and Sen Song. 2020. Unsupervised paraphrasing by simulated annealing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 302–312, Online. Association for Computational Linguistics.

Robin Lorenz, Anna Pearson, Konstantinos Meichanetzidis, Dimitri Kartsaklis, and Bob Coecke. 2021. Qnlp in practice: Running compositional models of meaning on a quantum computer.

Konstantinos Meichanetzidis, Alexis Toumi, Giovanni de Felice, and Bob Coecke. 2020. Grammar-aware question-answering on quantum computers.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. 2019. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6834–6842.

Eduardo Reck Miranda, Richie Yeung, Anna Pearson, Konstantinos Meichanetzidis, and Bob Coecke. 2021. A quantum natural language processing approach to musical intelligence.

Michael A Nielsen and Isaac Chuang. 2002. Quantum computation and quantum information.

Edwin Pednault, John A. Gunnels, Giacomo Nannicini, Lior Horesh, and Robert Wisnieff. 2019. Leveraging Secondary Storage to Simulate Deep 54-qubit Sycamore Circuits. ArXiv:1910.09534.

Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. 2014. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5(1):1–7.

John Preskill. 2018. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79.

Ananya B. Sai, Akash Kumar Mohankumar, and Mitesh M. Khapra. 2020. A survey of evaluation metrics used for nlg systems.

Peter W. Shor. 1997. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509.

Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan. 2020. t|ket⟩: a retargetable compiler for nisq devices. *Quantum Science and Technology*, 6(1):014003.

J.C. Spall. 1998. Implementation of the simultaneous perturbation algorithm for stochastic optimization. *IEEE Transactions on Aerospace and Electronic Systems*, 34(3):817–823.

James R. Wootton. 2020a. Procedural generation using quantum computation. FDG '20, New York, NY, USA. Association for Computing Machinery.

James R. Wootton. 2020b. A quantum procedure for map generation. In *2020 IEEE Conference on Games (CoG)*, pages 73–80.

Richie Yeung and Dimitri Kartsaklis. 2021. A ccg-based version of the discocat framework.

Daiwei Zhu, Gregory D Kahanamoku-Meyer, Laura Lewis, Crystal Noel, Or Katz, Bahaa Harraz, Qingfeng Wang, Andrew Risinger, Lei Feng, Debopriyo Biswas, et al. 2021. Interactive protocols for classically-verifiable quantum advantage. *arXiv preprint arXiv:2112.05156*.