Collected notes on topics in quantum computing

Rajas Chari

1 Semantics of Quantum protocols

In this section we briefly describe the literature surrokunding the use of quanutm computers to study semantics, and its applications in natural language processing(NLP)[1]. This follows the work of Coecke, who in 2010 first explored the use of closed compact categories to combine symbolic and distributional semantics to understand meanings of sentences from words in Ref. [1]. This was followed by a paper in which he explores the use of quantum algorithms to study compositional NLP in Ref. [2]. And finally, the real-world applications of these ideas were materialized by using the theory to implement qauntum algorithms on near-term quantum computers in Ref. [3].

1.1 Mathematical foundations for categorical semantics

The theory for understanding the meanng of sentences from the meaning of words has historically been divided into two classes:

1. the symbolic class: where the semantics is compositional, and the aim is to understand meaningns of compositions of words in a qualitative manner. An example of this we will refer to is the use of pregoups to undestand compositional logic of senteces.

2. the distributional class: where the understanding is uantitative and not compositions. Ans example we will be interested in is the representation of sentences with the *same* grammatical stucture as elements in a vector space. And then we can understand how close two sentences are in meaning by some inner product defined on this space. Note however, that we do not have a way of comparing two sentences which do not have the same grammatical structure. And we will try to solve this problem by lifting this theory to categories.

We descibe now the basic idea in Ref. [1] in which we claim that the two approaches above can be combined in a categorical semantics of sentences. Here, we will use monoidal categories where we will have *objects* as vector spaces which encode the meanings of words and sentences, and *morphisms* which we will use to encode the grammatical strucure of the sentences, or compositions. This method combines the compositional power of the method of pregroups, and the quantitative power of representations of meanings in vector spaces. In addition to this, we use the *tensor product* of the monoidal category to provide a bridge between vector spaces of different types, and to compare sentences within them, thus, overcoming the obstacle in canonical theories of distributional semantics.

[1]Not that we aim to describe very basic compositional semantics, which is essential for but not an important topic in modern frontiers in NLP based on neural networks. The potential value of this work will be to supplement efforts on the ML side by providing a better understanding of *context* using compositional semantics to understand sentences.

1.1.1 Closed compact categories

We assume formal knowledge of moonoidal categories and only present the bare minimum in the following definition:

Definition 1 (Monoidal category) *is a category, satisfying all the usual axioms of category theory, with a monoidal tensor product operation:*

$$\otimes : (A,B) \mapsto A \otimes B \tag{1}$$

A monoidal category i closed compact for each object A if exist objects A^r and A^l, and morphisms:

$$\eta^l : I \to A \otimes A^l, \quad \varepsilon^l : A^l \otimes A \to I \quad \eta^r : I \to A^r \otimes A \quad \varepsilon^r : A \otimes A^r \to I \tag{2}$$

this structure tells us that not all tensor products can be trivialized. In the graphical language, topology represents when interaction occurs, in other words, connectednedd encodes correlations. This is required for words like verbs, for example, which are of the form $(n^r s n^l) \in N \otimes S \otimes N$, and related to the objects and subjects components. This composite object cannot be separated into individual componenets because then it loses context. It is well knwn that pregrouops as posetal sets and vector spaces can be viewed as closed compact categories. And we use this fact to induce a closed compact structure on their product below.

1.1.2 The basic model

$$\begin{tikzcd}
 & language \arrow[dl, "meaning"'] \arrow[d] \arrow[dr, "grammar"] & \\
\mathbf{FVect} & \mathbf{FVect} \times P \arrow[l, "\pi_m"] \arrow[r, "\pi_g"'] & P
\end{tikzcd}$$

Figure 1

Explicitly, FVect $\times P$ is the category which has pairs (V,a) with V a vector space and $a \in P$ a grammatical type as objects, and the following pairs as morphisms:

$$(f : V \to W, p \leq q),$$

which we can also write as

$$(f, \leq) : (V,p) \to (W,q),$$

Note that if $p \leq q$ then there are no morphisms of type $(V,p) \to (W,q)$. It is easy to verify that the compact closed structure of FVect and P lifts componentwise to one on FVect $\times P$. The structural morphisms in this new category are now:

$$\begin{array}{ll}
(\eta^l, \leq) : (\mathbb{R}, 1) \to \left(V \otimes V, p \cdot p^l\right) & (\eta^r, \leq) : (\mathbb{R}, 1) \to (V \otimes V, p^r \cdot p) \\
\left(\varepsilon^l, \leq\right) : \left(V \otimes V, p^l \cdot p\right) \to (\mathbb{R}, 1) & (\varepsilon^r, \leq) : (V \otimes V, p \cdot p^r) \to (\mathbb{R}, 1)
\end{array}$$

1.1.3 Examples

The first example we study is the statement: *John likes Mary*. We encode this information as:

$$John \in V \quad likes \in T \quad Mary \in W \tag{3}$$

where V corresponds to the vector space of men, and W corresponds to the vector space of women. Elements $m_i \in V$ correspond to an instance of the class *men*, for example. Let *John* be element $m_3 \in V$, and *Mary* be element $f_6 \in W$. We then consider what the output or sentence space of this sentence is modelled. Let us consider a decision problem, where we want to decide if this is true of false, so we assign the zero vector for false, and unit vector to true. The trnasitive verb *likes* is encoded as a supeposition:

$$likes = \sum_{ij} m_i \otimes likes_{ij} \otimes f_j \tag{4}$$

where $likes_{ij} = 1$ if m_i likes f_j, and 0 if not. Then,

$$\begin{aligned} f(m_3 \otimes likes \otimes f_4) &= \sum_{ij} \langle m_3 | m_i \rangle likes_{ij} \langle f_j | f_4 \rangle \\ &= \sum_{ij} \delta_{3i} likes_{ij} \delta_{j4} \\ &= likes_{34} \end{aligned}$$

Here, the positive sentence with a trnasitive verb has a pregroup of type $n(n^r s n^l)n$. We assume meaning space of nouns are atomic: (V,n) and (W,n). And the meaning space of the verb is compound: $(V \otimes S \otimes W, n^r s n^l)$. The final map then has the form:

$$(V \otimes (V \otimes S \otimes W) \otimes W, n(n^r s n^l) n) \xrightarrow{(f,\leq)} (S,s) \tag{5}$$

and has the following synctactic reduction map:

$$f = \varepsilon_V \otimes 1_S \otimes \varepsilon_W : V \otimes (V \otimes S \otimes W) \otimes W \to S \tag{6}$$

2 Toric code

In this section we present the Toric code and discuss some aspects of quantum error correction and the stabilizer formalism which are apparent in this model. Notes are based on a given by Kitaev himself, in Google Quantum AI symposium 2022, see link.

2.1 The model

Consider a square lattice $\Lambda \subset \mathbb{R}^2$ made of L^2 sites. We can then decorate this lattice with qubits on the links of this lattice, with continuous boundary conditions, giving is $n = 2L^2$ qubits on a Torus. We then define the *Stabilizer operators* as:

$$A_s =_{j \in s} \sigma_j^x; \quad B_p =_{j \in \partial p} \sigma_j^z \tag{7}$$

where s is a vertex on the lattice and p is a plaquette. We can then define the quantum code space, or the space in which our logical qubits will reside, as the space of states which are stabilized by the above stabilizer operators, whcih we will denote as $\mathscr{L}$:

$$|\zeta\rangle \in \mathscr{L} \iff A_s|\zeta\rangle = B_p|\zeta\rangle = |\zeta\rangle \ \forall \ s,p \tag{8}$$

We will see later that the code space will be generated by Wilson loop generators defined on the non-trivial cocycles of the Torus, denotes $W_\alpha(\gamma) = \prod_\gamma \sigma^\alpha \ \ \forall \gamma \in \pi_1(\mathbb{T}^2)$, and will thus have dimension $\dim(\mathscr{L}) = 4$.

2.2 Error correction

2.2.1 The basics

Any error operator, denoted E, can be represneted in terms of a product of Pauli operators:

$$E = \sigma^x(g^x)\sigma^y(g^y) \tag{9}$$

where g^x and g^y are some sets of qubits. If any pauli operator acts on a logical state E defined by a staibilizer code with stabilizers S_i, for $i \in \{1,2,\dots l\}$, then we can compute the syndrome of the pauli operator by measuring all stabilizers of the code. A measurement will return -1 if the error acts ona n odd number of qubits on the domain of a stabilizer:

$$ES_i|\zeta\rangle = (-1)^{\mu_i} S_i E|\zeta\rangle \tag{10}$$

where the vector μ_i is called the *syndrome* of the error. Note that upon the action of the error E, the new state satisfies the new staibilizer conditions:

$$S_i|\psi\rangle = (-1)^{\mu_i}|\psi\rangle \tag{11}$$

The Pauli operator E can then fall into one of three categories:

1. it can be a trivial error, in the sense that it is a product of stabilizer operators, and therefore has a trivial syndrome, and does not change the logical state of the qubit.
2. it is a detectable error, in which case it has a non-trivial syndrome, but this at least leads us into a way to correct the error through some error correction scheme.
3. it is an undetectable error, i.e. $\mu_i = 0$ and $E \neq \prod S_i$. In this case, the operator preserves the code but not the individual code words. These are called logical operators, but if they occur spontaneously in the form of noise it is not good.

Note that logical operators are gates which can be used in our quantum circuits. And the errors of the second kind are errors which take us out of the code space, and thus cause what in technical jargon is called *leakage*.

2.2.2 Application to toric code: taking a page from simplicial homology

Applying this picture to the toric code, in which case errors can be thought of as spin flips or phases flips on chains of the lattice. Here, we mean chains in a very precise topological sense according to the definition below.

Definition 2 (Chain group) *Let $\{\sigma_i^\alpha\}$ denote a set of i-cells defined on the lattice Λ, then the chain group $C_i(\Lambda)$ is defined as the following $\mathbb{Z}$-graded group:*

$$C_i(\Lambda) = \bigoplus_\alpha \mathbb{Z}\sigma_i^\alpha \tag{12}$$

here a chain is just some element of the chain group, and we hope the definition is clear from context. Let us first treat only a single type of error, say bit flip error. We can treat the lattice itself as a valid CW-complex, with the errors corresponding to elements of the chain group $C_1(\Lambda)$. In this sense, it is easy to categorize the different types of errors in an efficient manner:

1. the trivial errors will corrspspond operators defined on 1-chains which are boundaries of some 2-chains, ie. $E \in B_1(\Lambda)$, where $B_1(\Lambda)$ is the $\mathbb{Z}$-graded abelain group of 1-dimensioanl boundaries of 2-dimensional elements of $C_2(\Lambda)$.

2. the deterctable errors will be operators defined on open chains, thus they will be $E \in C_1(\Lambda)/Z_1(\Lambda)$, where $Z_1(\Lambda)$ are cycles which don't have boundaries by definintion.

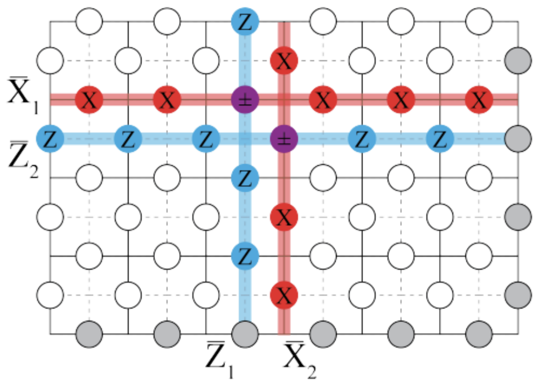
3. and finally the undetectable errors will correspond to operators defined on non-trivial elements of the 1st homology group, i.e. $E \in \pi_1(\Lambda) = Z_1(\Lambda)/B_1(\Lambda)$.

Note that this identification is complete! And this can be seen from the identity:

$$C_1(\Lambda) = C_1(\Lambda)/Z_1(\Lambda) \oplus Z_1(\Lambda)/B_1(\Lambda) \oplus B_1(\Lambda) \tag{13}$$

2.2.3 Logical qubits and the code space

Given a stabilizer code, by which we mean given the set of stabilizer operators and the Hilbert space they act on, we can define logical operators E to be operators which commute with all stabilizer operators, and thus preserve the code. It is easy to show that the upto trivial operators, the logical operators form a group generated by Z_1, X_1, Z_1, X_1 which are strings of σ^x and σ^z operators along the x, y directions respectively.



Furthermore, we observe that pairs of the above operators anti-commute with each other, and thus we can define a logical code space of complex dimension 2, or real dimension 4. We also define the code distance as the minimum size of a non-trivial logical operator, which in this case is the size of the torus $= L$.

Hamiltonian formalism

It is more natural and intuitive to describe the toric code in terms of a Hamiltonian given by:

$$H = -J_{charge}\sum_{s} A_s - J_{vortex}\sum_{p} + \text{perturbations} \tag{14}$$

where the ground state subspace corresponds to the code space, and error operators are spontaneous creation and annihilation operators of pairs of charges or vortices. Information int he GS subspace is *protected* from suffieciently small, constant or slowly varying adiabatic perturbations. Note that the Hamiltonain realization does not correct error, but rather, avoids errors. This is provided we are in equilibrium, the environment is cold, and abiabaticity, which implies high-frequency noise is absent. This is different from *active* error correction which we will describe next.

2.2.4 Error inference

In practice, if we are working with a stabilizeer code, we will at measured intervals of time do error correction in order to mantain proper functioning of our protocols. Durring this error correction routine, we will first measure the syndrome, upon which if there are errors we will see the location of the charges[2]. We then *infer* the errors that caused the charges, and this process is summarized below:

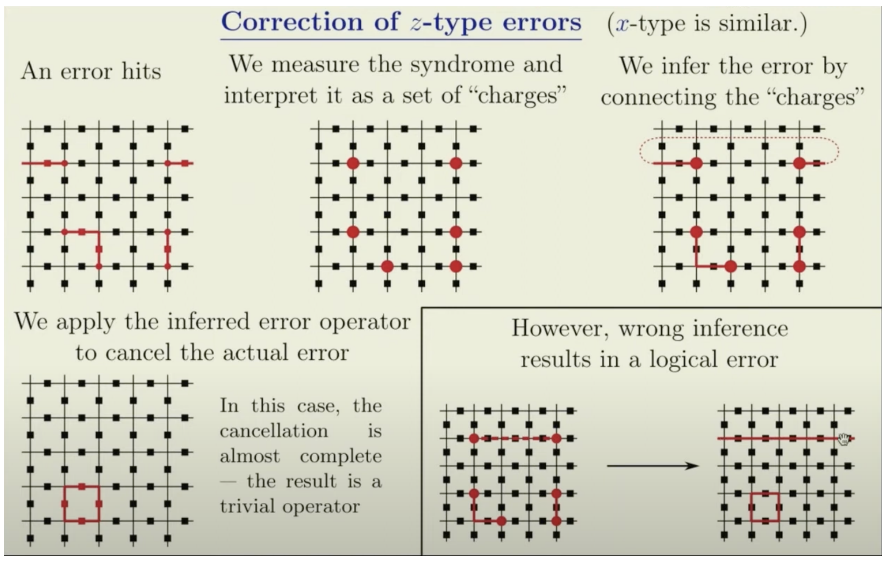

Correction of z-type errors
(x-type is similar.)
An error hits
We measure the syndrome and interpret it as a set of "charges"
We infer the error by connecting the "charges"
We apply the inferred error operator to cancel the actual error
In this case, the cancellation is almost complete — the result is a trivial operator
However, wrong inference results in a logical error

Figure 2: Inference protocols.

There exists multiple algorithms for erro inference/decoding which try to minimize the occurences of logical errors in error correction, the more popular ones are:

1. minimum weight matching: connect "charges" in pairs such that the inferred error has minimum weight.
2. maximum likelihood decoding: this is the optimal algorithm, in which errors with a given syndrome fall into four topological classes. Error inference paths which differe by a trivial

[2]Let us focus on z-type errors called charges. The formailsm extends trivially to vortices.

operator are said to be equivalent. We assign probabilities to each individual errors, and then calculate which error class is most likely. Any error from the most likely class is then selected.

The following theorem also makes it clear that in the thermodynamic limit these error correction algorithms make the code asymptotically fault-tolerant:

Theorem 3 (Threshold theorem) *Let each qubit undergo an x or z error with probabilities less than some threshold, denoted p_c, thne the maximum likelyhood decoding is asymptotically efficient:*

$$\varepsilon_L := Pr[\textit{logical error in an } L\times L \textit{ code}] \sim e^{-\alpha L}L \to 0 \quad as \ L\to\infty \tag{15}$$

These decoding schemes have some futher caveats which we mention:

1. faster, but less accurate, algorithms have slightly lower thresholds
2. measurement errors decrease the threshold further, and require 3D error tracking in space-time
3. imperfections in the implementation of measurements produce multi-qubit errors. Leading to *much* lower thresholds.
4. Good news: errors need not be fixed, but canbe tracked, and the decoding point can be postponed until a decision point.

2.3 Surface codes with boundary

The simplest planar codes are surface code with boundaries, and were first discussed in Ref. [4]. They have two logical operators, and hence only one logical qubit. In the schemeatic drawing, we see that we have rough boundaries and smooth boundaries. The rough boundaries can be used to absorb charges, and hence trivialize X operators along the x-direction, and the smooth boundaries can similirly absorb vortices:

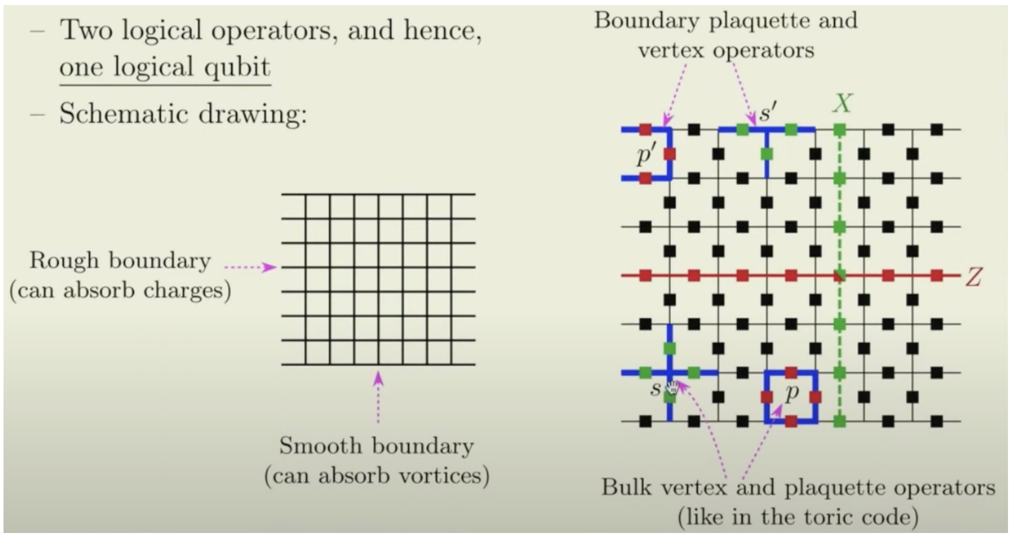

– Two logical operators, and hence, one logical qubit
– Schematic drawing:
Rough boundary (can absorb charges)
Smooth boundary (can absorb vortices)
Boundary plaquette and vertex operators
p'
s'
X
Z
s
p
Bulk vertex and plaquette operators (like in the toric code)

Figure 3: Surface codes with boundaries.

We have a small number of changes compared to the normal toric code: we have boundary stabilizer operators, and only two generators for the logical operators. The most economical of these codes, however, have diagonally oriented boundaries. Like the one used in Google's paper, Ref. [5].

2.4 The future

The standard fault-tolerant computation scheme, in the context of surface codes, should have the following ingrideients:

1. qubit initialization
2. measurement in the standard basis
3. depending on measurement results, we caan perform classical subroutines. Just before this step is a decision pint.
4. Clifford unitary gates, which will implement logical operaators.
5. It turns out the Clifford untiaries are not universal, due to which we need to introduce magic ancilla qubits.
6. implementing Clifford unitaries: trnasversal CNOT and semi-transversal Hadamard gates rewuire coupling or moving physical qubits over large distances. An aternate way to implement these gates is *lattice surgery*.

Ideas on implementing clifford unitaries:

1. Lattice surgery: See Ref. [6].
2. Using lattice disclination, and twists. See Ref. [7] .

3 Quantum machine learning(QML)

In this section, we will discuss some popular QML algorithms, and then we will put into context some of the problems I worked on during hackathons. In the process, I want to emphasize some generalities in the concepts which were missed when I first worked on these problems, real-time during the hackathons.

3.1 Quantum approximation optimization algorithm

Here, we will look at a heuristic explanation of the QAOA algorithm by looking at a simple example. Consider the Hamiltonian defined on three qubits:

$$H_{\text{cost}} = -J\{\sigma_1^z + \sigma_2^z + \sigma_{z3}\}(16)$$

The GS is clearly $|111\rangle$ state. Let us assume, for arguments sake, that we did knot know what the GS was, and it was our aim to find such a state using a quantum circuit. We can do this by following the protocol:

1. initiaalize the state in an equal superposition of the logical basis states: $|\psi_0\rangle = (\prod_i H_i)|0\rangle^{\otimes n}$, where H is the Hadamard operator.

2. apply a QAOA step, where we evolve the system over n layers as follows:

$$|\psi_n\rangle = \exp(-i\alpha_n H_{\text{mixer}})\exp(-i\gamma_n H_{\text{cost}})|\psi_{n-1}\rangle \tag{17}$$

   where each layer has evolutions with teo Hamiltonian, the cost Hamiltonian and the Mixer Hamiltonian, denoted H_{mixer}, where the mixer Hamiltonian is chosen such that it has sufficient expressibility in the Hilber space.[3] A single such step has $2n$ paraameters, denoted $\{\gamma_i, \alpha_i\}$.

3. We then pass the result to a ML subroutine, like gradient descent for example, which computes an acquisition function, which in this case will be:

$$f(\{\alpha_i, \gamma_i\}) = \langle\psi_n|H_{\text{cost}}|\psi_n\rangle \tag{18}$$

   The ML subroutine then feeds back parameters in order to minimize the acquisition function in a loop, until we pass a certain threshold.

This protocol is the basic idea behind which the QAOA algorithm works. We note that the algorithm is only a heuristic and approximate algorithm. So, we make no claims about solving a given problem, and consequently no claims about complexity of problems.

3.1.1 Some observations

Observing that the input to a QAOA mostly involves just knowing the cost Hamiltonian, if we can re-package a given problem in the form of finding the GS energy of a quantum Hamiltonian, then we can apply QAOA to solve the problem. This of course comes with mutiple caveats:

1. all problems might not admit a simple local Hmailtonian which describes the peculiarities of the problem.

2. even in the class of problems that admit a local Hamiltonian description, the problem of finding this Hamiltonian description might also be *hard*, in the complexity theory sense.

3. given a problem and its description of a problem, a QAOA implementation for a solution only proves a heuristic method to solve a problem. In addition to this, it will likely be slower than most classical heuristic algorithms given hardware restrictions. However, everything is not lost. Scaling analysis might prove that the algorithm scaales well, and will eventually beat a classical computer.

4. since the algorithm is heuristic, we cannot make any complexity theoretic claims. What we mean by this is that the numebr of iterations that will be required for an exact solution are simply unknown.

Here, we note an interesting pattern which we see quite often in our study of QCQI: given a classical problem, we try to find a quantum Hamiltonaian which encodes the problem. Being able to do this is already a motivating sigm to reach a quantum solutiont o a given problem. For example,

[3]By this we mean, we want the mixer to take any state into a superposition of states which is dense in the Hilbert space.

in Ref. [8], the authors are able to find a map[4] between the famous Sherrington-Kirkpatrick model and the MaxCut problem. The map is given below:

$$C(x) = \frac{1}{2}\sum_{ij\in E}(1 - s_i s_j), \quad s_i \in \{\pm 1\}; \iff H = \frac{1}{2}\sum_{ij\in E}(I - Z_i Z_j) \tag{19}$$

MaxCut is a paradigmatic example, but there exist other known mappings to quantum hamiltonians. For example, Boolean functions can be mapped to quantum hamiltonains.

3.2 Variational quantum algorithms

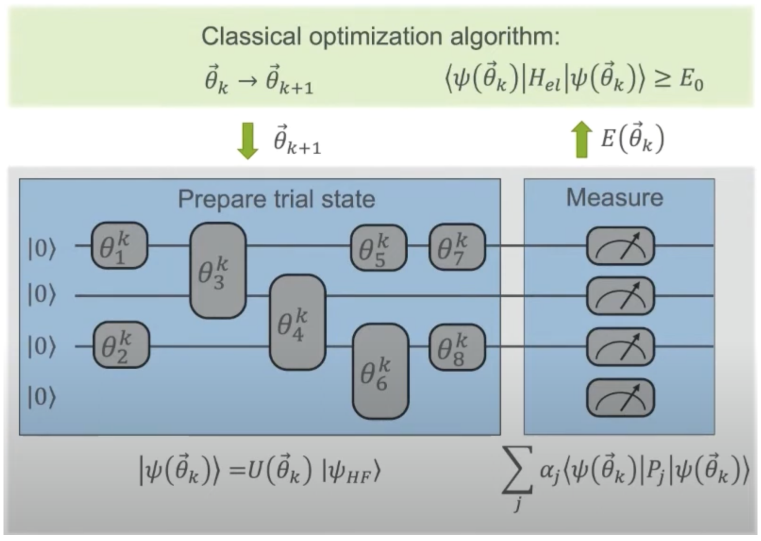
We are going to discuss the variational quantum eigensolver(VQE) algorithm from the perspective of applications to solve problems in quantum chemistry. This should serve the dual purpose of presenting both the method and its potential application. VQEs are hybrid quantum algorithm, which are based on the variational principle in QM. Basically we have a quantum Hamiltonian which encodes a given problem, and a classical subroutine which constructs a variational quantum state in the Hilbert space associated to this Hamiltonian. The aim is to find the GS of the Hamiltonian, and we do this by following a feedback protocol very similar to gradient descent. Here, the uantum subrouting calculates the acquisition function, and the classical subroutine generates the variational state depending on the answer.

This is easier to understand in terms of the example of solving molecular dynamics problems. It is well known that the HF ansatz gives a classical heuristic method to solve for the GS of a molecular Hamiltonian. But we propose an alternate solution. We map the fermionic hamiltonian to a spin Hamiltonian, and generate an ansatz:

$$|\psi(\theta_i)\rangle = U(\theta_i)|\psi_{HF}\rangle \tag{20}$$

here the parameters $\{\theta_i\}$ are the variational parameters, which we will optimiz to improve the HF solution. Then we follow the protocol described above: calculate the acquisition function in the quanutm module, pass the results to the ML module and improve the guess and repeat, as shown in the figure below:

[4]not sure if the map is faithful



Classical optimization algorithm:
$\vec{\theta}_k \rightarrow \vec{\theta}_{k+1}$
$\langle\psi(\vec{\theta}_k)|H_{el}|\psi(\vec{\theta}_k)\rangle \geq E_0$
$\vec{\theta}_{k+1}$
$E(\vec{\theta}_k)$
Prepare trial state
Measure
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
θ_1^k
θ_2^k
θ_3^k
θ_4^k
θ_5^k
θ_6^k
θ_7^k
θ_8^k
$|\psi(\vec{\theta}_k)\rangle = U(\vec{\theta}_k)\,|\psi_{HF}\rangle$
$\sum_j \alpha_j \langle\psi(\vec{\theta}_k)|P_j|\psi(\vec{\theta}_k)\rangle$

Figure 4: VQE protocol.

Appendix A BigQ Hackathon preparation notes

For the final report, see BigQ folder on local machine, or public notes on github.

A.1 Introduction

The purpose of these notes is to document what I learned in the weeks before and during the BigQ quantum computing Hackathon held in Chicago in Fall of 2023. The details of the event can be found here. The architecture we will be using during the hackathon was not revieled to us until the event, so I figured I should guess what it would be before so I can get a head start. The obvious choice was PASQAL, which was a startup and one of the major sponsors of the event.

A.2 PASQAL: neutral atom technology

PASQAL uses neutral atom technology to simulate quantum networks. The two broad features of the technology seem to be the following:

1. using optical tweezers to isolate and arrange the atoms into arrays which form the quantum network. The distances between the atoms can be controlled, and this will trun out to be important when we discuss interactions through the Rydberg channel.
2. using the concept of Rydberg blockade to simulate effects of interactions among the atoms. The mechanism is very simple, almost like hard-core bosons, where two atoms cannot be excited into the Rydberg states if they are too close together. Here too close means a distance less than the Rydberg radius.

A.3 Atomic states

Very breifly, the atom seems to have two regimes of states:

1. the **ground state** seems to be a non-degenerate set made of two states, labelled $|0\rangle$ and $|1\rangle$. The transitions between these are called **hyperfine** transtions, and are driven by the **Raman channel**. So, the Raman channel can drive transitions that look like: $|0\rangle \longleftrightarrow |1\rangle$.
2. and then we have the hyper-**excited** states, called the Rydberg states, which are highly excited($n \sim 50$) states which have a large wavefunction radius, and thus large polarizability making them susceptible to interactions. The resonances are contrlled such that only one of the Rydberg states is accessible and is called our excited state, denotes $|r\rangle$.

Driving transitions to the Rydberg states is a bit more subtle. We note that if two atoms are too close to each other **and** both are in the $|1\rangle$ state, ie. $|11\rangle$, then they acnnot be driven to the excited state $|rr\rangle$. In short, we can say $|11\rangle \not\longrightarrow |rr\rangle$. But we can drive all other ground states to the excited states: $|00\rangle$, $|01\rangle$, $|10\rangle \longrightarrow |rr\rangle$, and this mechanism can be exploited to implement **controlled** gates.

A.4 Methods

We summarize braod category of methods used in quantum computing in this section.

A.4.1 Mapping to Ising models

The problem to solve for the ground state of an Ising model is well known to be efficiently done on a quantum computer. Refer to Clark course notes and code. The ubiquity of application and natural mappings of real-life problems to the Ising model makes this problem of quintessential importance. In essense, a great subset of network problems, which are not hard to come by in nature, can be well approximated by a random Ising model given by:

$$H[J_{ij}, H_i] = \sum_{ij} J_{ij} \sigma^i \sigma^j + \sum_i h_i \sigma^i \tag{21}$$

where J_{ij} can be understood to represent random interactions amoung nodes, which can be drawn from a given distribution for example. And h_i can represent external probes an forces acting on the nodes. The ground state represents the minimal energy state of the energy functional. General methodology to solve the problem would be to use Monte Carlo simulations.

A.4.2 Use cases

1. Power grid dynamics
2. Traveling salesman problem: dilevery, transit, logistic problem: dilevery, transit. Use vaaitional quantm eigensolver

References

[1] Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. *Mathematical Foundations for a Compositional Distributional Model of Meaning*. 2010. arXiv: 1003.4394 [cs.CL].

[2] William Zeng and Bob Coecke. "Quantum Algorithms for Compositional Natural Language Processing". In: *Electronic Proceedings in Theoretical Computer Science* 221 (Aug. 2016), pp. 67–75. ISSN: 2075-2180. DOI: 10.4204/eptcs.221.8. URL: http://dx.doi.org/10.4204/EPTCS.221.8.

[3] Konstantinos Meichanetzidis et al. "Quantum Natural Language Processing on Near-Term Quantum Computers". In: *Electronic Proceedings in Theoretical Computer Science* 340 (Sept. 2021), pp. 213–229. ISSN: 2075-2180. DOI: 10.4204/eptcs.340.11. URL: http://dx.doi.org/10.4204/EPTCS.340.11.

[4] S. B. Bravyi and A. Yu. Kitaev. *Quantum codes on a lattice with boundary*. 1998. arXiv: quant-ph/9811052 [quant-ph].

[5] Rajeev Acharya et al. "Suppressing quantum errors by scaling a surface code logical qubit". In: *Nature* 614.7949 (Feb. 2023), pp. 676–681. ISSN: 1476-4687. DOI: 10.1038/s41586-022-05434-1. URL: https://doi.org/10.1038/s41586-022-05434-1.

[6] Dominic Horsman et al. "Surface code quantum computing by lattice surgery". In: *New Journal of Physics* 14.12 (Dec. 2012), p. 123011. DOI: 10.1088/1367-2630/14/12/123011. URL: https://dx.doi.org/10.1088/1367-2630/14/12/123011.

[7] H. Bombin. "Topological Order with a Twist: Ising Anyons from an Abelian Model". In: *Phys. Rev. Lett.* 105 (3 July 2010), p. 030403. DOI: 10.1103/PhysRevLett.105.030403. URL: https://link.aps.org/doi/10.1103/PhysRevLett.105.030403.

[8] Joao Basso et al. “The Quantum Approximate Optimization Algorithm at High Depth for MaxCut on Large-Girth Regular Graphs and the Sherrington-Kirkpatrick Model”. en. In: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. DOI: 10.4230/LIPICS.TQC.2022.7. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.TQC.2022.7.