Near Term Advances in Quantum Natural Language Processing

Dominic Widdows, Daiwei Zhu, Chase Zimmerman
IonQ, Inc. 4505 Campus Drive, College Park, MD

Quantum NLP Goals

Holy Grails and Cornucopias

Building real Artificial General Intelligence (AGI) is a core goal of computer science. There are reasons for believing that quantum models and quantum computing will fill crucial roles in truly intelligent systems.

IonQ is already investing in this ambitious goal, asking:
1. What will AGI really need? What necessary properties are lacking today?
2. What advantages might quantum systems offer?
3. What quantum systems today convincingly demonstrate proof-of-concept?
4. What core mathematical / language operations can be identified?
5. For each operation, what are the simplest effective implementations?

These questions help to motivate medium-term challenges and immediate progress.

Example Area: Ambiguity in Language

1. AGI will need to be aware of many possible goals, intents, actions, across many modalities at once, and to recognize which few are relevant now.
2. Quantum systems offer the ability to maintain exponentially many states - even if most are never activated or observed!
3. There are several methods in NLP that have convincingly demonstrated that ambiguous words can be represented and manipulated using vectors, matrices, tensor products, and projections. Are these naturally *quantum* systems?
4. If so, vectors and linear operators are key mathematical ingredients.
5. What would be a simple effective implementation of this?

Let's string some techniques together and start building something!

Vectors Can Represent Ambiguity

"A vector is a single point and so cannot represent something comprised of many parts such as a polysemous / ambiguous word." *Misconception in NLP literature (2012-ongoing)*

In quantum mechanics, state vectors are introduced *because* they can represent superpositions of many ingredients - so this mistake doesn't arise!

Ambiguous words in some models are naturally learned as sums of vectors representing particular senses. For example: $\overrightarrow{\mathsf{Java}} = \overrightarrow{\mathsf{Java}}_{\mathsf{island}} + \overrightarrow{\mathsf{Java}}_{\mathsf{programming}}$

Matrices Can Resolve Ambiguity

Matrix multiplication is an operation used for composition that sometimes resolves ambiguity. In this toy domain, "*visit(X)*" maps a Place X to an Event.

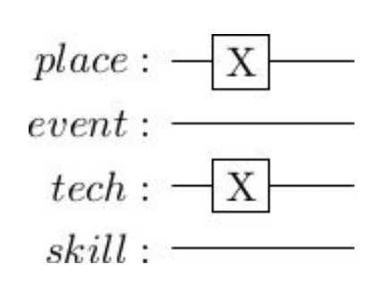
	Borneo	Java	C++	Trip	Juggling
Place	1	1	0	0	0
Event	0	0	0	1	0
Tech	0	1	1	0	0
Skill	0	0	0	0	1

$$\mathsf{visit} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \mathsf{learn} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

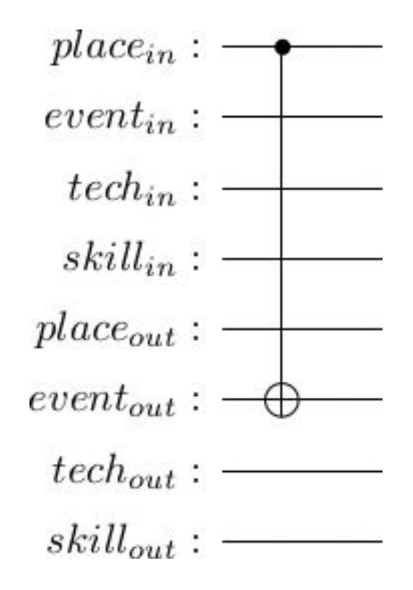
$$\mathsf{visit(Borneo)} = \mathsf{visit(Java)} = \mathsf{Event}$$
$$\mathsf{learn(C{+}{+})} = \mathsf{learn(Java)} = \mathsf{Skill}$$

Unwanted senses of "Java" aren't considered and rejected - they're just ignored. That's handy - but the matrices aren't unitary gates. So what can we do instead?

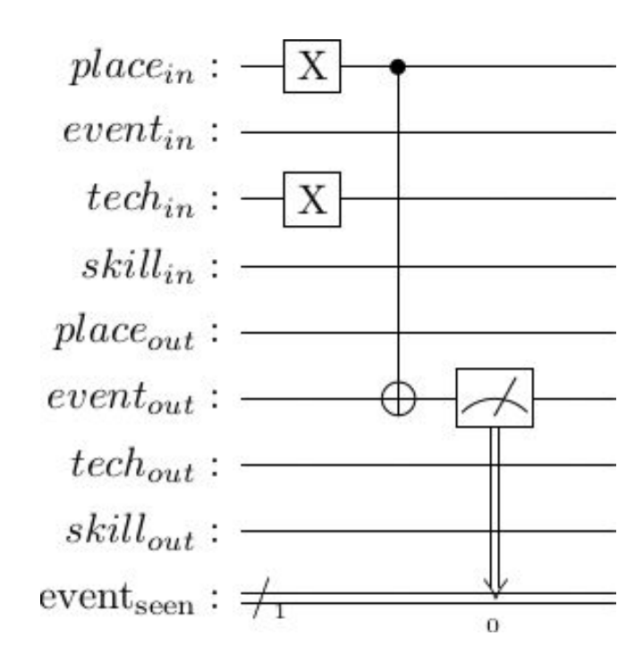
Java - "place" or "tech"



Visit - maps "place" to "event"



Visit Java is an "event"



This works, but doesn't scale. Can the same behavior be achieved with fewer qubits? Must the mapping from qubits to topics be stored externally in classical memory?

Quantum Demonstration Systems

Bigram Model Using Quantum Probability

This mathematics is drawn from Bradley's *Statistical Algebra* (2020) and Formal Concept Analysis. Understanding when pairs of words mean similar things is important in NLP. For example, verb-noun pairs are sometimes clustered to identify "intents" in dialog systems.

Verb Noun Examples	Adjective Noun Examples (Used Below)
Book room, arrange accommodation, reserve hotel, make reservation, …	green apple, red apple, yellow banana, ripe banana, red pepper, yellow pepper, black shoes, red dress, blue suit, white shirt.

Note that the individual words aren't always synonyms - for example, *book ≠ make*.

Modelling Steps

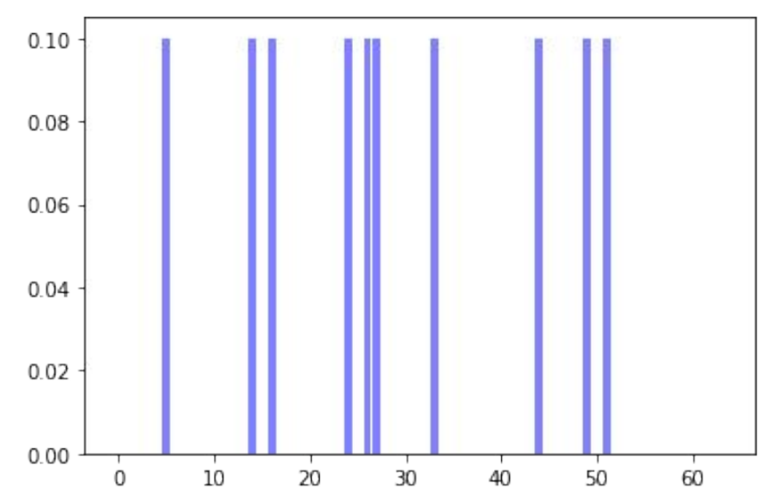
- Collect example pairs, e.g.. "red apple", "green apple", "yellow banana".
- Encode as a state vector $|\psi\rangle$ in the tensor product space
- Partial traces of the density operator $|\psi\rangle\langle\psi|$ encode information
  - Main diagonal: Individual probabilities
  - Off diagonal: Pairwise correlations

$$\begin{array}{c|ccc} & \text{red} & \text{green} & \text{yellow} \\ \hline \text{red} & \frac{1}{3} & \frac{1}{3} & 0 \\ \text{green} & \frac{1}{3} & \frac{1}{3} & 0 \\ \text{yellow} & 0 & 0 & \frac{1}{3} \end{array}$$

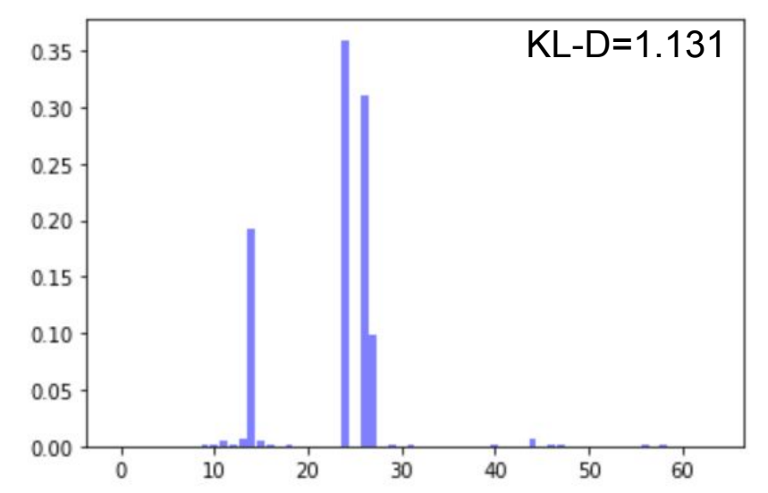
$$|\psi\rangle = \frac{1}{\sqrt{\sum P(a,b)^2}} \sum P(a,b)\, |a\rangle \otimes |b\rangle$$

Quantum Generative Model

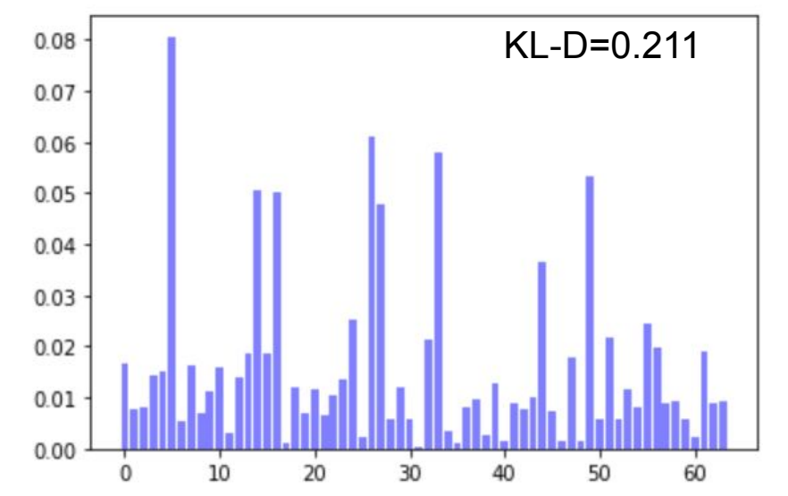
- Give each adj, noun an index, e.g., *blue* $|101\rangle$ *shoes* = $|100\rangle$
  - Concatenate to form bigram index, *blue shoes* $|101100\rangle$ (see bottom right)
- Use these to form state $|\psi\rangle$ and variational circuit trained to model bigram distributions
- Best results starting with some noise - like smoothing in language models



Target Distribution



Model trained without noise (mode collapsing behavior)

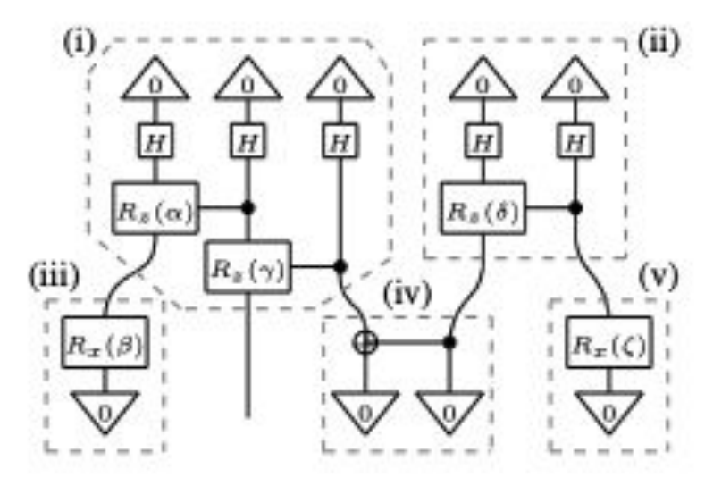
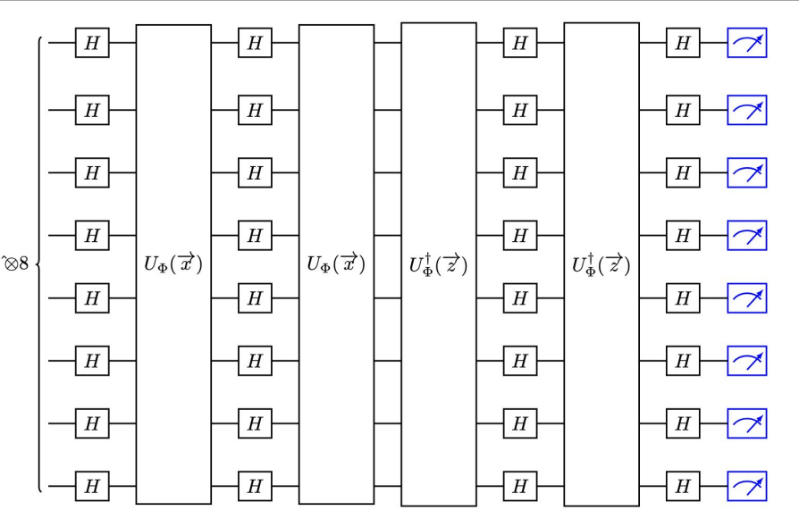
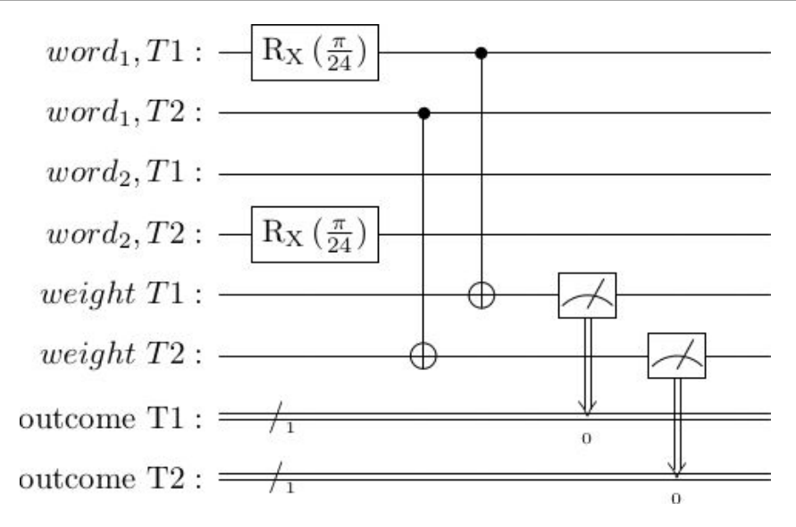


Model trained with noise

Quantum SVM and Bag-of-Words Classifiers

Classifying texts to topics is a standard NLP challenge. An example was performed on quantum hardware by Lorenz et al *QNLP in Practice* (2021). Here we extend this work with alternatives.

Dataset generated for this purpose:
- 70 training phrases, 30 test phrases, marked with a topic: *"skillful woman runs application"* ⇒ Computing, *"man prepares tasty dinner"* ⇒ Food
- Published with *lambeq* package (with thanks to those authors!).

DisCoCat Model (Lorenz, Coecke et al)	**Quantum-Enhanced Support Vector Machine** (this work)	**Quantum Bag-of-Words Classifier** (this work)
Uses Categorial Grammar with classical parser, creates circuit ansatz for nouns, verbs, adjectives, creates tensor network.	Uses classical Word2Vec embeddings (top right) to encode words as feature vectors. Trains and runs QSVM for classification.	Encodes each (word, topic) weight learned from training in a qubit. Classical state maps words / topics to qubits. Classified with quantum adder circuit (right).
		
Runs on 6 qubit machine.	Runs on 8 qubit machine.	Runs on 9 qubit QC (per topic) or 18 qubit simulator (single run).
Accuracy: 83.3%	Accuracy: 90%	Accuracy: 100%

This example shows that we can build successful quantum classifiers, and compare models for accuracy, sophistication, complexity, and hardware cost. Such design tradeoffs will become normal.

Basic Building Blocks

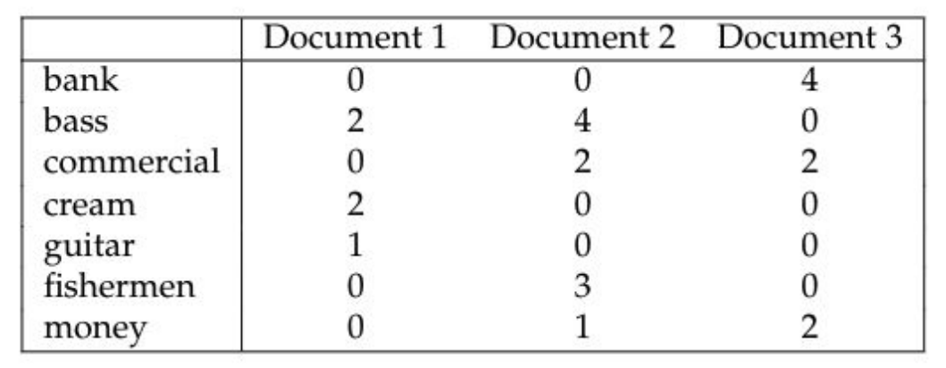
Distributional Semantic Vectors

"Die Bedeutung eines Wortes ist sein Gebrauch in der Sprache." (Mostly!)
Wittgenstein, Philosophical Investigations §43 (1953)

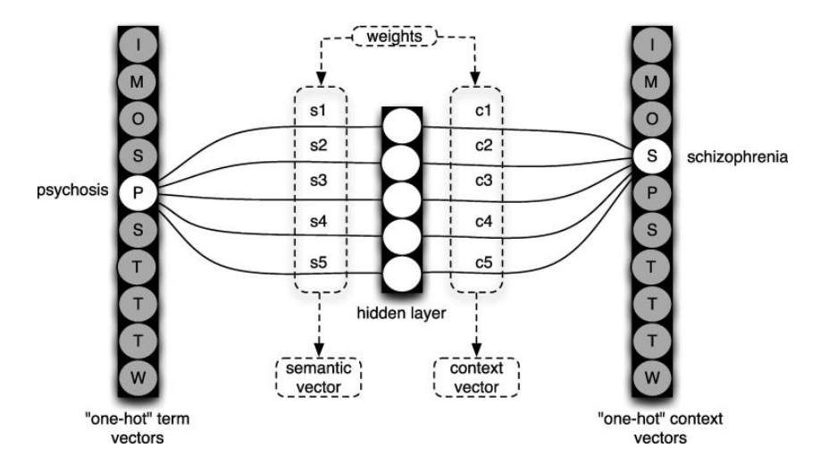
"*don't be such an* ___. You shall know a word by the company it keeps!"
JR Firth, A Synopsis of Linguistic Theory (1957)

Vectors in Language Systems

- Term Document Matrices
  - Information Retrieval since 1960s
  - Term vectors and Doc vectors
- Projection using SVD, NNMR, LDA
  - Latent Semantic Analysis, Topic Maps
- Connectionist Models / Neural Nets$_{\varphi\varphi}$
  - Since 1980s, concepts, roles, bindings
  - Word2Vec Embeddings (2013)
  - ELMo, BERT, GPT "contextual" models



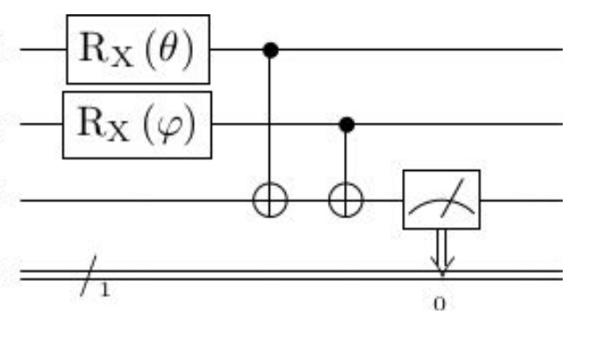
	Document 1	Document 2	Document 3
bank	0	0	4
bass	2	4	0
commercial	0	2	2
cream	2	0	0
guitar	1	0	0
fishermen	0	3	0
money	0	1	2



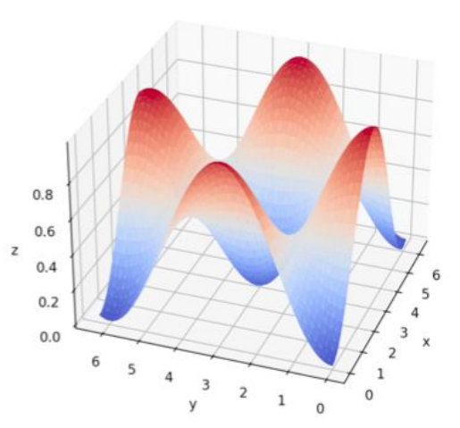
Addition/Combination of Inputs

From the summation of term vectors to make document vectors (1960s) to the deep neural networks and activation functions of today, there are situations where several contributions needs to be gathered into some sort of "additive" combination.

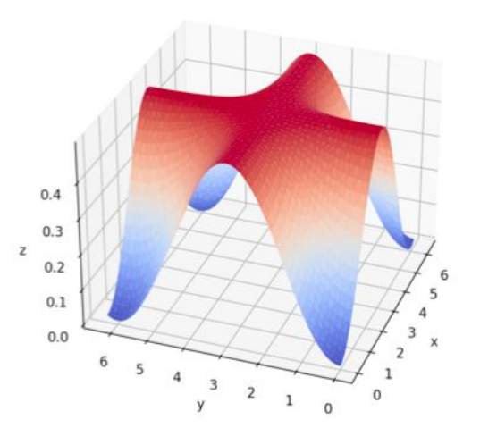
Adder components using fractional X-rotations (right) were used in the bag of words classifier experiments.



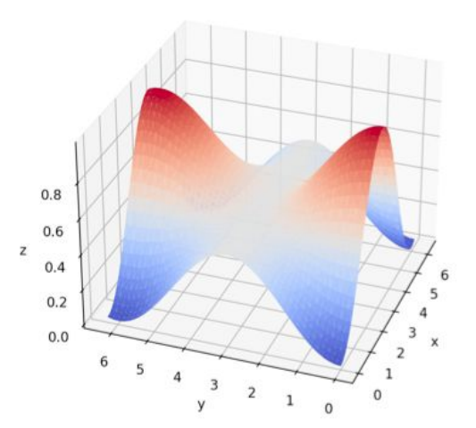
$$P(1) = \cos^2(\theta)\sin^2(\varphi) + \sin^2(\theta)\cos^2(\varphi)$$



Fractional X rotations



Fractional Hadamards



Mixture of X and Hadamard

Different Fractional Ingredients Lead to Different Combination Properties

Medium Term Challenges

Storing / accessing state

Language competence requires a lot of information. The distributional patterns and correlations between words that we need to know to use them effectively can make language models grow very big. Building thoroughly quantum language models is one of the many motivating use-cases for qRAM. In the meantime, on quantum computers we often work with small artificial datasets, provided the belief that these methods could scale up with suitable hardware support is well-motivated.

Representing the Quantum Bigram Model

The quantum bigram model (center) is an example of the dense encoding enabled by quantum systems. For a joint distribution with *n* prefixes and *m* suffixes, we would need *nm* classical bits just to model "present or absent". For the quantum model, we can use $\lceil \log_2(n) \rceil + \lceil \log_2(m) \rceil$ qubits, and an encoding convention where each prefix and suffix is given a binary index, and the cooccurrence weight is stored in the state given by concatenating these two indices. Creating and reading an arbitrary superposition of such states is not yet simple!

Stability through computations combining many signals

Quantum errors can propagate through systems. It's important to track which information is relatively stable in the presence of small errors (such as the difference between 1.0001 and 1.0002 as numbers) and which information is entirely changed by small errors (such as the difference between "*botch*" and "*batch*").

References
Scan the QR code, follow the hyperlinks!

qip 2022

IONQ