Quantum Machine Intelligence (2024) 6:60
https://doi.org/10.1007/s42484-024-00193-w

RESEARCH ARTICLE

Large scale structure-aware pronoun resolution using quantum natural language processing

Hadi Wazni[1] · Kin Ian Lo[1] · Lachlan McPheat[1] · Mehrnoosh Sadrzadeh[1]

Received: 30 June 2023 / Accepted: 19 August 2024

Abstract
Natural language consists of words, sentences, and larger units. Guided by grammatical structure, words compose with each other to form sentences. Similarly, guided by discourse structure, sentences compose with each other to form dialogues and documents. Classical machine learning algorithms have achieved significant success in learning the meanings of words. When it comes to meanings of sentences and discourse units, they however fall short of being compositional. The DisCoCat model of meaning, introduced by Clark, Coecke, and Sadrzadeh in 2010, provides a solution—at the sentence level—using higher-order tensors, the learning of which has been a challenge. A recent initiative known as Quantum Natural Language Processing (QNLP) introduces a translation between the DisCoCat tensors and Variational Quantum Circuits (VQC). This offers the potential of learning these higher-order tensors more efficiently when the circuits are executed on quantum computers. In previous work, we lifted the DisCoCat framework from the sentence level to the discourse level using a Fock space semantics. In this paper, we extend the DisCoCat-VQC translation to this semantics and experiment with it in a discourse task. We develop a massive dataset with 16,400 entries inspired by a major coreference resolution task, known as the Winograd Schema Challenge, proposed as a test of machine intelligence. Noisy and noiseless simulations were executed on IBMQ software, and the parameters of the discourse VQCs were learnt. The model converged to 77.5% accuracy, surpassing a Bag-of-Words model that neglects any structure. It also outperformed 2 out of 3 state-of-the-art classical coreference resolution architectures. These findings highlight the significant potential of quantum machine learning in advancing discourse analysis and structured natural language processing.

Keywords Coreference resolution · Quantum natural language processing · Hybrid quantum-classical methods · Variational quantum circuits · Quantum machine learning · Categorial grammars · Modal Lambek calculus · Tensor space semantics · Diagrammatic calculi · Fock spaces

1 Introduction

The seminal work of Lambek in 1958 (Lambek 1958) showed that the simple logic of concatenation and its residuals form a *Syntactic Calculus*. This calculus, referred to as the *Lambek Calculus*, could model and reason about grammatical structures of sentences of natural language. The atomic formulae of the logic model basic grammatical types, e.g. noun phrases n and declarative sentences s. Concatenations of formulae model compositions of types.

Subsequent work of Moortgat in 1996 (Moortgat 1996), Jaeger in 1998 (Jäger 1998), Morrill in 2015, 2016 (Morrill and Valentín 2015, 2016), and Kanovich et al. in 2016, 2020 (Kanovich et al. 2016, 2020) have expanded the expressive power of the Lambek Calculus by adding modalities. Moortgat used these modalities to restrict associativity, Morrill was in favour of using them for island types and iterative conjunctives, and Kanovitch et al. focused on parasitic gaps. While these models primarily focused on sentence-level analysis, Jaeger's work extended the application of the Lambek Calculus to the discourse level, enabling the modelling of

✉ Hadi Wazni
hadi.wazni.20@ucl.ac.uk

Kin Ian Lo
kin.lo.20@ucl.ac.uk

Lachlan McPheat
l.mcpheat@ucl.ac.uk

Mehrnoosh Sadrzadeh
m.sadrzadeh@ucl.ac.uk

1 University College London, London, UK

Published online: 16 September 2024

relationships between multiple sentences. An example of such a relationship is coreference, that is, when two or more expressions in a text or discourse refer to the same entity or concept. This can include cases where a pronoun refers back to a previously mentioned noun phrase (pronoun coreference or anaphora), as well as cases where an expression is omitted but understood based on the context (ellipsis).

In previous work (McPheat et al. 2020), we combined Jaeger's idea with *Lambek Calculus with Soft Sub-Exponentials* introduced by Kanovitch et al. in Kanovich et al. (2020), since it had better meta logical properties, i.e. finite rules, cut elimination, strong normalisation, and decidability. We showed how it can model and reason about coreference relations such as pronoun resolution and ellipsis and developed a finite-dimensional vector spaces for it. This development was in the style of the vector space semantics of Lambek calculus (Sadrzadeh et al. 2013), but with a novel feature. The soft sub-exponentiated formulae were interpreted as *truncated Fock spaces*. A Fock space is the direct sum of all tensor powers of a vector space. Fixing the field to be $\mathbb{F}$, the Fock space over a vector space V is

$$\mathbb{F} \oplus V \oplus (V \otimes V) \oplus (V \otimes V \otimes V) \oplus \cdots$$

A truncated version of this space is obtained restricting the Fock space to its k_0'th tensor power, for k_0 a fixed bound, defined as follows:

$$\mathbb{F} \oplus V \oplus (V \otimes V) \oplus (V \otimes V \otimes V) \oplus \cdots \oplus \underbrace{(V \otimes V \otimes \cdots \otimes V)}_{k_0} .$$

The bound k_0 is fixed by the logic. The soft sub-exponentiated formulae of the logic $!A$ can be thought of as *storages of formulae*, which only contain k_0 copies of a formula A. Access to these copies is obtained by a ! elimination rule in the logic and by projection to the desired level of a truncated Fock space in the semantics. When modelling coreference relations such as pronoun resolution, only a fixed number of pronouns refer to a noun phrase in any given discourse. An upper bound can easily be determined from these fixed numbers, e.g. by averaging.

Historically, Lambek Calculus has relied on relational semantics and a question arises that 'what is the benefit of working with a vector space semantics?'. The answer comes from recent advances in Natural Language Processing, where vector semantics are learnable via machine learning algorithms such as neural networks. Having a vector space semantics for Lambek Calculus and its modal extensions enables us to work with machine-learned vector representations of words, sentences, and discourse units in a structured manner. However, these semantics often rely on computationally expensive higher-order tensors. In response, Quantum Natural Language Processing (QNLP) research proposes leveraging quantum computers, which can handle tensors more efficiently. It computes word embeddings as parameterised quantum circuits that can solve NLP tasks faster than any classical computer (Zeng and Coecke 2016; Wiebe 2019). It is inspired by categorical quantum mechanics and the DisCoCat framework, making use of string diagrams to translate from grammatical structure to quantum processes (Coecke et al. 2020). The studies in Ma and Tresp (2021); Ma et al. (2019) also explore learning higher-order tensors derived from knowledge graphs via variational quantum circuits.

In order to take advantage of QNLP, we develop a string-diagrammatic calculus for the truncated Fock space semantics of our logic and use the open source libraries *DisCoPy* (de Felice et al. 2021) and *Lambeq* (Kartsaklis et al. 2021) to generate corresponding quantum circuits. We learn the parameters of these circuits on a definite pronoun resolution task inspired by the Winograd Schema Challenge (Levesque et al. 2012; Rahman and Ng 2012). This challenge was designed to assess the ability of natural language processing systems to comprehend ambiguous pronouns and resolve referential dependencies. It consists of a set of carefully crafted sentences designed to be difficult for AI systems that rely solely on statistical patterns or syntactic analysis.

Achieving high accuracy necessitates training the variational circuits on a massive dataset. To generate this dataset, we adopted a few-shot prompting technique using the GPT−3.5 large language model (Brown et al. 2020). We expanded an initial set of sentence pairs from Rahman and Ng (2012) to include a total of 16,400 pairs. These pairs were then divided into training, validation, and test subsets according to a 60-20-20 split. We treated the problem as a classification task and ran the variational quantum circuits on the IBMQ simulators. The circuits derived from our logic take the full grammatical and discourse structures into account. For comparison, we also generated quantum circuits for a Bag-of-Words model, where neither grammatical nor discourse structures are accounted for. Both models converged during training, with the model incorporating discourse and grammatical structures achieving a higher accuracy.

We benchmarked our model against three classical coreference resolution architectures—CoreNLP, Neural Coreference, and SpanBERT—none of which explicitly encode sentence grammatical structure. Our model outperformed CoreNLP and Neural Coreference with an average accuracy improvement of 37.2%. However, SpanBERT surpassed the performance of our model by a margin of 7.00%.

The paper is organised around four core sections. Section 2 presents the type-logical grammar **SLLM** which allows referencing. Section 3 introduces a vector-space semantics of

SLLM that makes use of Fock spaces. Section 4 introduces a diagrammatic calculus to reason about **SLLM** using string diagrams. Section 5 describes how string diagrams can be converted to parametrised quantum circuits using quantum ansätze. Experiments and concluding remarks are presented in Sects. 6 and 7.

2 Lambek calculus with soft subexponentials (SLLM) and its applications to modelling discourse structure

The formulae of Lambek calculus with soft Subexponentials, **SLLM**, are generated using the following BNF:

$$A, B ::= A \in At \mid A \cdot B \mid A\backslash B \mid A/B \mid !A \mid \nabla A$$

The set of atoms can be any set of indices. For the linguistic application purposes of this paper, we set the atoms to be $\{s, n\}$ with s representing the grammatical 'declarative sentence' and n the 'noun phrase'. With the above BNF over this set of atoms, we can generate the usual set of complex types, including the formula for an adjective n/n, the formula for an intransitive verb $n\backslash s$, and the formula for a transitive verb $n\backslash s/n$.

Given the types of the calculus, we define its *sequents* to be pairs, written $\Gamma \longrightarrow A$, where Γ is a finite list of formulas, $\Gamma = A_1, A_2, \ldots, A_n$ and A is a formula. Given this notion of sequents, we define the logical and structural rules of **SLLM** in a Gentzen calculus style below.

In linguistic applications, Γ often represents the grammatical types of a string of words, and A is the result of the composition of these types. An example of a grammatical rule of English is that a noun phrase, such as *John*, composes with an intransitive verb such as *sleeps* to form a sentence, that is *John sleeps*. This is modelled in **SLLM** via sequent $n, n\backslash s \longrightarrow s$, and its proof:

$$\dfrac{\dfrac{}{n \longrightarrow n} \quad \dfrac{}{s \longrightarrow s}}{n, n\backslash s \longrightarrow s}\,\backslash L \tag{1}$$

The modality ! is known as a soft sub-exponential. The banged formulae are thought of as *storages* and the *!*-modality itself as a *projection* operation. Reading $!_L$-structural rule from bottom to top, we are projecting from the storage *!A* which contains k_0 formulae A, to n copies of A, for $1 \leq n \leq k_0$. This allows for the existence of a controlled implicit notion of copying in the syntax, which is not the same as the usual on-the-nose notion of copying: we are not replacing $!A$ with copies of *!A* nor are we replacing A with copies of A. We are just projecting from a storage containing many copies of A to a smaller number of those copies. This implicit notion of copying comes from Soft Linear Logic (Lafont 2004) has a neat interpretation in the vector space semantics as we show in 3. The second modality, ∇, is the one that allows its formulas to be permuted. This is seen in the rules *perm* and *perm'* in Table 1.

The bounding of n is strictly necessary, as having an unbounded number of copies makes the calculus' derivability problem undecidable. This bound is not a problem in terms of modelling language, as the bound corresponds to the number of times one may refer to something in a discourse. A bad but symbolic bound then would be the number of words in the discourse you want to analyse.

The way we model discourse phenomena such as pronoun resolution and ellipsis is similar to that of Jäger (1998, 2006). In Jäger's work, the word that is being referred to is explicitly copied and moved to the site of the referring word, where it gets applied. In our framework, however, we do not have explicit copying and use a storage type for the word that is being referred to. For example, if we wish to model the discourse *John sleeps. He snores.*, we need to first assign a storage type to *John*, then project two copies from it, and move one to the site of the type of *He*. Then apply the type of *He* to the moved copy of *John*. So we assign the type $!\nabla n$ to *John*, and assign the type $\nabla n\backslash n$ to the pronoun *He*. This latter represents a function which takes a projected-from-a-storage noun as its (left)input and returns a noun as its outputs. This assignment is summarised below:

$$\{(John : !\nabla n), (sleeps : n\backslash s), (He : \nabla n\backslash n), (snores : n\backslash s)\}.$$

The discourse structure of *John sleeps. He snores.* is modelled by the following **SLLM** proof tree:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{n \longrightarrow n} \quad \dfrac{\dfrac{\dfrac{}{n \longrightarrow n}}{\nabla n \longrightarrow \nabla n} \quad \dfrac{\dfrac{}{n \longrightarrow n} \quad \dfrac{\dfrac{}{s \longrightarrow s} \quad \dfrac{}{s \longrightarrow s}}{s, s \longrightarrow s, s}\,\cdot R}{s, n, n\backslash s \longrightarrow s, s}\,\backslash L}{s,\ \nabla n,\ \nabla n\backslash n, n\backslash s \longrightarrow s, s}\,\backslash L}{n, n\backslash s,\ \nabla n,\ \nabla n\backslash n, n\backslash s \longrightarrow s, s}\,\backslash L}{\nabla n, n\backslash s,\ \nabla n,\ \nabla n\backslash n, n\backslash s \longrightarrow s, s}\,\nabla_L}{\nabla n,\ \nabla n, n\backslash s,\ \nabla n\backslash n, n\backslash s \longrightarrow s, s}\,perm}{!\nabla n, n\backslash s,\ \nabla n\backslash n, n\backslash s \longrightarrow s, s}\,!L$$

Table 1 Sequent presentation of **SLLM**

$\dfrac{}{A \longrightarrow A}\,I$	
$\dfrac{\Gamma \longrightarrow A \quad \Sigma_1, B, \Sigma_2 \longrightarrow C}{\Sigma_1, \Gamma, A\backslash B, \Sigma_2 \longrightarrow C}\,\backslash L$	$\dfrac{A, \Gamma \longrightarrow B}{\Gamma \longrightarrow A\backslash B}\,\backslash R$
$\dfrac{\Gamma \longrightarrow A \Sigma_1, B, \Sigma_2 \longrightarrow C}{\Sigma_1, B/A, \Gamma, \Sigma_2 \longrightarrow C}\,/L$	$\dfrac{\Gamma, A \longrightarrow B}{\Gamma \longrightarrow B/A}\,/R$
$\dfrac{\Gamma_1, A, B, \Gamma_2 \longrightarrow C}{\Gamma_1, A \cdot B, \Gamma_2 \longrightarrow C}\,\cdot L$	$\dfrac{\Gamma_1 \longrightarrow A \quad \Gamma_2 \longrightarrow B}{\Gamma_1, \Gamma_2 \longrightarrow A \cdot B}\,\cdot R$
$\dfrac{\Gamma_1, \overbrace{A,\ A,\ \ldots,\ A}^{n \text{ times}}, \Gamma_2 \longrightarrow B}{\Gamma_1, !A, \Gamma_2 \longrightarrow B}\,!_L$	$\dfrac{A \longrightarrow B}{!A \longrightarrow !B}\,!_R$
$\dfrac{\Gamma_1, A, \Gamma_2 \longrightarrow B}{\Gamma_1, \nabla A, \Gamma_2 \longrightarrow B}\,\nabla_L$	$\dfrac{A \longrightarrow B}{\nabla A \longrightarrow \nabla B}\,\nabla_R$
$\dfrac{\Gamma_1, \Gamma_2, \nabla A, \Gamma_3 \longrightarrow B}{\Gamma_1, \nabla A, \Gamma_2, \Gamma_3 \longrightarrow B}\,perm$	$\dfrac{\Gamma_1, \nabla A, \Gamma_2, \Gamma_3 \longrightarrow B}{\Gamma_1, \Gamma_2, \nabla A, \Gamma_3 \longrightarrow B}\,perm'$

Reading the proof from bottom to top, we see that the application of $!_L$ replaces the type of *John* ($!\nabla n$) with two copies of *John* (∇n). The application of *perm* moves one of the copies next to *He*. The following $\backslash_L$ application identifies *He* with *John*, and the application of ∇_L 'forgets' that the other copy of *John* is a copy. The sequent above the application of ∇_L corresponds to the typing of *John sleeps. John snores.*, and the proof of it shows that this is indeed two sentences.

In order to show that the system can model more complex example, we consider the anaphoric reference: *The ball hit the window and Bill caught it*, directly from the pronoun resolution dataset (Rahman and Ng 2012). Here, *it* refers to *The ball*. For brevity, we may type the whole noun phrase *The ball* as $!\nabla n$, and the noun phrase *the window* as n[1], and as usual, we type *and* as $s\backslash s/s$ and both verbs *hit* and *caught* as $n\backslash s/n$. Finally, we type the pronoun *it* as $\nabla n\backslash n$.

Note that this example consists of only one sentence. Proof trees for two-sentence examples are shown in 3 and 4.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{n \longrightarrow n}{\nabla n \longrightarrow \nabla n} \quad \dfrac{\dfrac{n \longrightarrow n \quad \dfrac{n \longrightarrow n \quad \dfrac{s \longrightarrow s \quad \dfrac{n \longrightarrow n \quad \dfrac{n \longrightarrow n \quad \dfrac{s \longrightarrow s \quad s \longrightarrow s}{s/s, s \longrightarrow s}\,/_L}{s/s, s/n, n \longrightarrow s}\,/_L}{s/s, n, n\backslash s/n, n \longrightarrow s}\,\backslash_L}{s, s\backslash s/s, n, n\backslash s/n, n \longrightarrow s}\,\backslash_L}{s/n, n, s\backslash s/s, n, n\backslash s/n, n \longrightarrow s}\,/_L}{n, n\backslash s/n, n, s\backslash s/s, n, n\backslash s/n, n \longrightarrow s}\,\backslash_L}{\nabla n, n\backslash s/n, n, s\backslash s/s, n, n\backslash s/n, n \longrightarrow s}\,\nabla_L}{\nabla n, n\backslash s/n, n, s\backslash s/s, n, n\backslash s/n, \nabla n, \nabla n\backslash n \longrightarrow s}\,\backslash_L}{\nabla n, n\backslash s/n, n, s\backslash s/s, n, \nabla n, n\backslash s/n, \nabla n\backslash n \longrightarrow s}\,perm}{\nabla n, n\backslash s/n, n, s\backslash s/s, \nabla n, n, n\backslash s/n, \nabla n\backslash n \longrightarrow s}\,perm}{\nabla n, n\backslash s/n, n, \nabla n, s\backslash s/s, n, n\backslash s/n, \nabla n\backslash n \longrightarrow s}\,perm}{\nabla n, n\backslash s/n, \nabla n, n, s\backslash s/s, n, n\backslash s/n, \nabla n\backslash n \longrightarrow s}\,perm}{\nabla n, \nabla n, n\backslash s/n, n, s\backslash s/s, n, n\backslash s/n, \nabla n\backslash n \longrightarrow s}\,perm}{!\nabla n, n\backslash s/n, n, s\backslash s/s, n, n\backslash s/n, \nabla n\backslash n \longrightarrow s}\,!_L \tag{2}$$

The dog broke the vase. It was clumsy, which naturally has type $s \cdot s$ in the following proof:

$$\textit{the dog} :!\nabla n,\ \textit{broke} : n\backslash s/n,\ \textit{the vase} : n,\ \textit{It} : \nabla n\backslash n,$$
$$\textit{was} : n\backslash s/(n/n),\ \textit{clumsy} : n/n,$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{n \longrightarrow n}{\nabla n \longrightarrow \nabla n}\,\nabla_R \quad \dfrac{\dfrac{n \longrightarrow n \quad \dfrac{n \longrightarrow n \quad \dfrac{n \longrightarrow n \quad \dfrac{\dfrac{\dfrac{n \longrightarrow n \quad n \longrightarrow n}{n/n, n \longrightarrow n}\,/_L}{n/n \longrightarrow n/n}\,/_R \quad \dfrac{s \longrightarrow s \quad s \longrightarrow s}{s, s \longrightarrow s \cdot s}\,\cdot_R}{s, s/(n/n), n/n \longrightarrow s \cdot s}}{s, n, n\backslash s/(n/n), n/n \longrightarrow s \cdot s}\,\backslash_L}{s/n, n, n, n\backslash s/(n/n), n/n \longrightarrow s \cdot s}\,/_L}{n, n\backslash s/n, n, n, n\backslash s/(n/n), n/n \longrightarrow s \cdot s}\,\backslash_L}{\nabla n, n\backslash s/n, n, n, n\backslash s/(n/n), n/n \longrightarrow s \cdot s}\,\nabla_L}{\nabla n, n\backslash s/n, n, \nabla n, \nabla n\backslash n, n\backslash s/(n/n), n/n \longrightarrow s \cdot s}\,\backslash_L}{\nabla n, n\backslash s/n, \nabla n, n, \nabla n\backslash n, n\backslash s/(n/n), n/n \longrightarrow s \cdot s}\,perm}{\nabla n, \nabla n, n\backslash s/n, n, \nabla n\backslash n, n\backslash s/(n/n), n/n \longrightarrow s \cdot s}\,perm}{!\nabla n, n\backslash s/n, n, \nabla n\backslash n, n\backslash s/(n/n), n/n \longrightarrow s \cdot s}\,!_L \tag{3}$$

[1] The modelling can me made be more precise by typing all instances of *the* to n/n and typing *ball* to $!\nabla n$ and *window* to n.

We can resolve object anaphora, as in the example: *The cat broke the glass. It was fragile.*

the cat : n, *broke* : $n\backslash s/n$, *the glass* :$!\nabla n$, *It* : $\nabla n\backslash n$,

was : $n\backslash s/(n/n)$, *fragile* : n/n,

$$\dfrac{\dfrac{\overline{n \longrightarrow n}}{\nabla n \longrightarrow \nabla n}\,\nabla_R \qquad \dfrac{\dfrac{\overline{n \longrightarrow n} \qquad \dfrac{\overline{n \longrightarrow n} \qquad \dfrac{\overline{n \longrightarrow n} \qquad \dfrac{\dfrac{\dfrac{\overline{n \longrightarrow n} \quad \overline{n \longrightarrow n}}{n/n, n \longrightarrow n}\,/_L}{n/n \longrightarrow n/n}\,/_R \qquad \dfrac{\overline{s \longrightarrow s} \quad \overline{s \longrightarrow s}}{s, s \longrightarrow s\cdot s}\,\cdot_R}{s, s/(n/n), n/n \longrightarrow s\cdot s}\,/_L}{s, n, n\backslash s/(n/n), n/n \longrightarrow s\cdot s}\,\backslash_L}{s/n, n, n, n\backslash s/(n/n), n/n \longrightarrow s\cdot s}\,/_L}{n, n\backslash s/n, n, n, n\backslash s/(n/n), n/n \longrightarrow s\cdot s}\,\backslash_L}{n, n\backslash s/n, \nabla n, n, n\backslash s/(n/n), n/n \longrightarrow s\cdot s}\,\nabla_L}{\dfrac{n, n\backslash s/n, \nabla n, \nabla n, \nabla n\backslash n, n\backslash s/(n/n), n/n \longrightarrow s\cdot s}{n, n\backslash s/n, !\nabla n, \nabla n\backslash n, n\backslash s/(n/n), n/n \longrightarrow s\cdot s}\,!_L}\,\backslash_L \tag{4}$$

Ellipsis is modelled similarly. Since we have only experimented with anaphoric reference in the current paper, we refer the reader for the typing and proof tree of an example, e.g. *John plays guitar. Mary does too.* to previous work (McPheat et al. 2020).

3 Truncated Fock space semantics of SLLM

In this section, we recall the definition of the vector space semantics of **SLLM**, first defined in McPheat et al. (2020). We define the semantics inductively on formulas, which are interpreted as vector spaces and proofs, which are interpreted as linear maps. We will use a semantic bracket notation $[\![\ ]\!]$: **SLLM** $\to$ **FdVect** to denote the semantics of formulas $[\![A]\!]$ and the semantics of proofs $[\![\pi]\!]$, where if π is a proof of a sequent $\Gamma \longrightarrow A$, then $[\![\pi]\!]$ is a linear map from $[\![\Gamma]\!]$ to $[\![A]\!]$.

1. For the atomic formulas n, s we interpret them as some fixed vector spaces $N := [\![n]\!]$ and $S := [\![s]\!]$.
2. Formulas of the form $A\cdot B$ are interpreted using the tensor product $[\![A \cdot B]\!] := [\![A]\!] \otimes [\![B]\!]$.
3. Formulas of the form $A\backslash B$ are interpreted using the vector space dual $[\![A\backslash B]\!] := [\![A]\!]^* \otimes [\![B]\!]$. Similarly, we have $[\![B/A]\!] := [\![B]\!] \otimes [\![A]\!]^*$.
4. ∇-formulas are interpreted trivially $[\![\nabla A]\!] := [\![A]\!]$.
5. !-formulas are interpreted using truncated Fock-spaces $[\![!A]\!] := T_{k_0}[\![A]\!]$, where

$$T_{k_0}[\![A]\!] = \bigoplus_{i=0}^{k_0} [\![A]\!]^{\otimes i} = k \oplus [\![A]\!] \oplus ([\![A]\!]\otimes [\![A]\!]) \oplus ([\![A]\!] \otimes [\![A]\!] \otimes [\![A]\!]) \oplus \cdots \oplus [\![A]\!]^{\otimes k_0}.$$

As a notational convenience, we will write $[\![\Gamma]\!]$ for $[\![A_1]\!] \otimes [\![A_2]\!] \otimes \cdots \otimes [\![A_n]\!]$ for $\Gamma = A_1, \ldots, A_n$. The vector space semantics of proofs is as follows:

1. The axiom $\overline{A \longrightarrow A}$ is simply interpreted as the identity matrix $I_{[\![A]\!]} : [\![A]\!] \to [\![A]\!]$.
2. The $\cdot_L$ rule is trivially interpreted, as the semantics does not distinguish between the , and the $\cdot$.
3. The $\cdot_R$ rule is interpreted as the tensor product of linear maps. That is, given proofs of the hypotheses of the $\cdot_R$-rule π_1 of $\Gamma_1 \longrightarrow A$ and π_2 of $\Gamma_2 \longrightarrow B$, we have the semantics of the new proof, ending with the $\cdot_R$ rule as $[\![\pi_1]\!] \otimes [\![\pi_2]\!]$.
4. The $\backslash_L$ and $/_L$-rules are interpreted as application. That is, given proofs π of $\Gamma \longrightarrow A$ and τ of $\Sigma_1, B, \Sigma_2 \longrightarrow C$, we have the semantics of the proof ending with the $\backslash_L$-rule being

$$[\![\tau]\!]\circ(I_{[\![\Sigma_1]\!]}\otimes \mathrm{ev}^l_{[\![A]\!],[\![B]\!]}\otimes I_{[\![\Sigma_2]\!]})\circ(I_{[\![\Sigma_1]\!]}\otimes[\![\pi]\!]\otimes I_{[\![A\backslash B]\!]}\otimes I_{[\![\Sigma_2]\!]}).$$

   Similarly, for the $/_L$-rule with the same π and τ as above, we have that the semantics of the new proof, now ending with $/_L$ is

$$[\![\tau]\!]\circ(I_{[\![\Sigma_1]\!]}\otimes \mathrm{ev}^r_{[\![A]\!],[\![B]\!]}\otimes I_{[\![\Sigma_2]\!]})\circ(I_{[\![\Sigma_1]\!]}\otimes I_{[\![B/A]\!]}\otimes[\![\pi]\!]\otimes I_{[\![\Sigma_2]\!]}).$$

5. The $\backslash_R$ and $/_R$-rules are interpreted as currying. That is, given a proof π of $A, \Gamma \longrightarrow B$, the semantics of the proof beginning with π and ending with $\backslash_R$ is the curried version of $[\![\pi]\!]$, which we denote as

$$\Lambda^l[\![\pi]\!] : [\![\Gamma]\!] \to [\![A]\!] \otimes [\![B]\!].$$

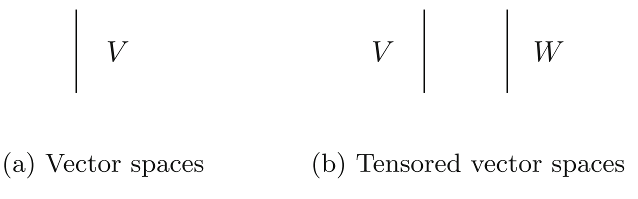

Fig. 1 Standard graphical calculus for vector spaces

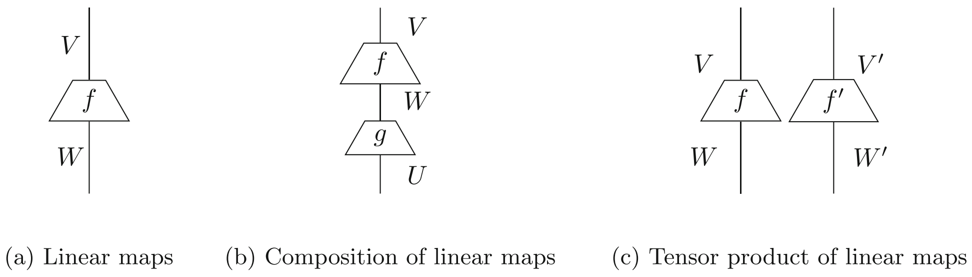
V
f
W
V
f
W
g
U
V
f
W
f′
V′
W′
(a) Linear maps
(b) Composition of linear maps
(c) Tensor product of linear maps

Fig. 2 Linear maps

Similarly for the $/_R$-rule, if we assume the sequent $\Gamma, A \longrightarrow B$ has proof τ, the new proof ending in $/_R$ has semantics

$$\Lambda_r[\![\tau]\!] : \Gamma \to [\![B]\!] \otimes [\![A]\!].$$

6. Both the ∇_L and ∇_R-rules are trivial, since we interpret ∇ trivially in the vector space semantics.
7. The *perm* and *perm*′-rules are interpreted using the symmetry, σ, of the tensor product. Recall that for any two vector spaces V, W, we have that $\sigma_{V,W} : V \otimes W \cong W \otimes V$. Using this symmetry, we can interpret a proof π followed by the *perm*-rule as $[\![\pi]\!] \circ (I_{[\![\Gamma_1]\!]} \otimes \sigma_{[\![A]\!],[\![\Gamma_2]\!]} \otimes I_{[\![\Gamma_3]\!]})$
8. The $!_L$-rule, the one that lets us copy, is interpreted using the projections from the truncated Fock space. Recall that whenever you have a direct sum $V \oplus W$ of vector spaces V, W, we have two canonical projections, namely $p_V : V \oplus W \to V$ and $p_W : V \oplus W \to W$ defined as $p_V(v, w) = v$ and $p_W(v, w) = w$. These projections extend to any number of summands, which in our case is k_0. Thus, if we consider a proof π of a sequent $\Gamma_1, A, A, \ldots, A, \Gamma_2 \longrightarrow B$ (with n instances of A, say) which is followed by an application of the $!_L$-rule, the semantics of the whole proof becomes

$$[\![\pi]\!] \circ (I_{[\![\Gamma_1]\!]} \otimes p_n \otimes I_{[\![\Gamma_2]\!]})$$

where $p_n : T_{k_0}[\![A]\!] \to [\![A]\!]^{\otimes n}$, is the nth projection map.
9. The $!_R$-rule is interpreted as the application of T_{k_0}. That is, given a proof π of the sequent $A \longrightarrow B$, which is followed by an application of $!_R$ to give $!A \longrightarrow !B$, the semantics of the whole proof is $T_{k_0}[\![\pi]\!]$. By this notation, we mean the following map:

$$(T_{k_0}[\![\pi]\!])(\bigotimes_{i=0}^{j} a_i^j)_{j=0}^{k_0}(\bigotimes_{i=0}^{j}[\![\pi]\!]a_i^j)_{j=0}^{k_0}$$

where a_0 is some element in the ground field, and $a_i^j \in [\![A]\!]$ for all $1 \leq i \leq j \leq k_0$. That $T_{k_0}[\![\pi]\!]$ yields a linear map is not obvious, but it is a well-known fact proven in any text on universal enveloping algebras, for instance (Humphreys 1972).

4 Diagrammatic computations

4.1 Existing string diagrammatic calculus

We recall the standard graphical language for finite dimensional vector spaces. Vector spaces are denoted by labelled strings, as in Fig. 1a, where we have drawn a vector space V. By convention, the 1-dimensional vector space is not drawn at all. The tensor product of two vector spaces is denoted by placing the corresponding strings side-by-side as in Fig. 1b, where we have drawn $V \otimes W$.

Linear maps $f : V \to W$ are denoted as boxes on strings, with their domain feeding into the box from above, and the codomain coming out from below, as in Fig. 2a. The composition of linear maps is denoted by vertical superposition of boxes, as in Fig. 2b, where we have drawn the composition

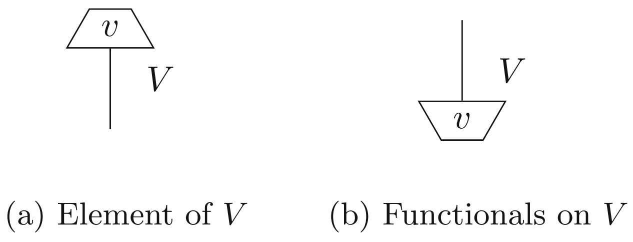

Fig. 3 Vectors and linear functionals

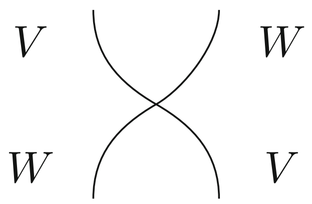

Fig. 4 Symmetry of tensor

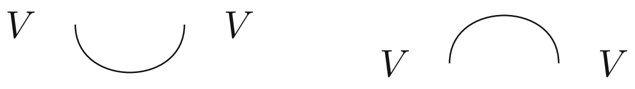

Fig. 5 Cups and caps

of maps $f : V \to W$ and $g : W \to U$. The tensor product of linear maps is depicted by horizontal juxtaposition, as in Fig. 2c.

Vectors $v \in V$ are in bijection with linear maps $k \to V$, so we may think of vectors as a special kind of linear functions. Since we do not draw the ground field at all, we may draw vectors as boxes with no output, as in Fig. 3a. We orient these boxes as we will need to manipulate diagrams with element-boxes, and in doing so, it is useful to keep track of which way round your diagram is drawn. Note that a linear functional, i.e. a linear map $V \to k$, is a box with no output, and so we may draw it as in Fig. 3b. We draw it with the opposite orientation to illustrate the self-duality of **FdVect**, where every vector $v \in V$ defines a functional via the inner product $(v, -) : V \to k$ which maps $w \mapsto (v, w)$.

The symmetry of the tensor product, i.e. $V \otimes W \cong W \otimes V$ for any vector spaces V, W, is drawn by crossing wires as in Fig. 4.

Inner products are depicted by cups, as drawn in Fig. 5. Dually, we have caps, which are maps $k \to V \otimes V$ corresponding to the unit map $1 \mapsto \sum_{i=1}^{n} v_i \otimes v_i$ where we take V to have basis $\{v_i\}_{i=1,...,n}$.

These diagrams satisfy the usual string diagrammatic equations, e.g. the most important of which is the following known as *yanking* in Fig. 6.

4.2 New diagrams for Fock spaces

Next, we introduce diagrams for our Fock spaces. We depict Fock spaces, i.e. vector spaces of the form $T_{k_0}V$ by bold strings, labelled with a V, see Fig. 7. The special diagrammatic structure we have on Fock spaces is the projection to the n-th layer, which is denoted by the usual linear map notation, which in this case we label by p_n, and call a p_n-box, as in Fig. 7.

Similar to vectors from vector spaces, vectors from Fock spaces $T_{k_0}V$ are depicted with linear maps $v : k \to T_{k_0}V$, see Fig. 8.

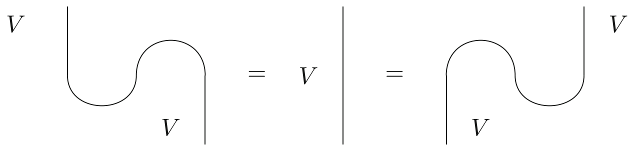

Fig. 6 Yanking

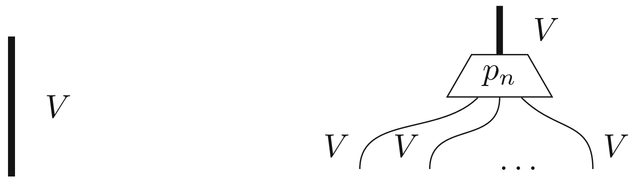

Fig. 7 Fock spaces and projections from them

A series of operations often performed on Fock spaces is accessing an element via the above linear map, followed by a projection to the n-th layer, see Fig. 9.
When modelling discourse phenomena that span multiple sentences, we need Fock spaces. Unfortunately, Fock spaces do not have direct counterparts in the standard model of quantum computation. To address this limitation, we observe that string diagrams involving Fock spaces can be *only* manipulated using a sequence of two operations: accessing an element of a Fock space followed by a projection to an n-th layer. To incorporate this into our quantum circuit representation, we replace this sequence of operations with an order n tensor, as depicted in Fig. 10. We refer to this replacement as a *Fock Space Projection* step. Formally speaking, by doing so, we are restricting ourselves to only the n-th layer of a Fock space, which is nothing but an order n tensor, for which we have a quantum circuit counterpart in IBMQ.

5 From string diagrams to quantum circuits

In Lorenz et al. (2021), a four-step process is described to represent a sentence as a parameterised quantum circuit. We adapt these steps to represent a discourse as a quantum circuit, which are as follows:

1. Discourse parsing: The first step involves parsing the given discourse into a proof in **SLLM**, using the *BobCatParser* (Clark 2021; Yeung and Kartsaklis 2021). This process involves analysing the grammatical structure of the text and extracting relevant linguistic components.

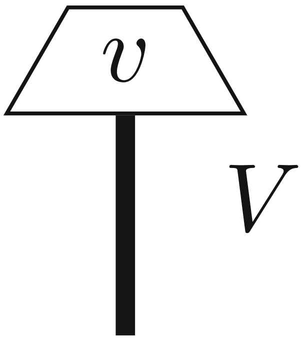

Fig. 8 Elements of Fock spaces

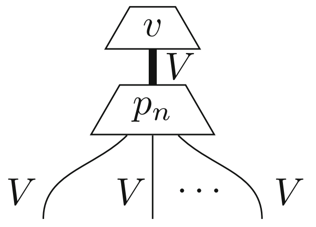

Fig. 9 Accessing an element of Fock spaces followed by a projection

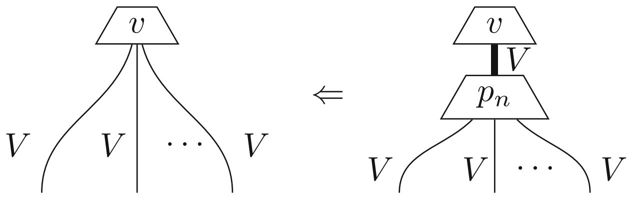

Fig. 10 Simplification step to encode the Fock space as an IBMQ quantum circuit

2. String diagram generation: After parsing the discourse, we transform the resulting proof tree into a string diagram, which visually depicts the relationships and connections between the linguistic elements in the text. We employ *DisCoPy* (de Felice et al. 2021) as a backend for generating and processing these diagrams.
3. String diagram simplification: The generated string diagram is subjected to simplification using a built-in rewrite rule in *Lambeq* (Kartsaklis et al. 2021), which substitutes every determiner with a cap (see Fig. 5), effectively removing it from the diagram after wire stretching. The diagram then undergoes the Fock Space Projection simplification, described above and depicted in Fig. 10, which is new to this paper and specific to Fock spaces. These simplifications are preformed to reduce the complexity of the variational quantum circuits.
4. Diagram normalisation: Following the simplification step, the simplified string diagram undergoes normalisation. This process involves removing cups (representing interactions), stretching wires (indicating connections), and rearranging boxes (depicting linguistic elements). These transformations are performed to optimise the diagram for efficient execution on quantum computers.
5. Quantum circuit transformation: The resulting simplified and normalised diagrams are transformed into a quantum circuit. This conversion is based on a parameterization scheme, i.e. an ansatz.[2] In this work, we chose to use the *Instantaneous Quantum Polynomial* (IQP) ansatz, which consists of interleaving layers of Hadamard quantum gates with diagonal unitaries (Shepherd and Bremner 2009; Havlíček et al. 2019).

We elaborate on the Quantum Circuit Transformation step above. To do so, we follow the standard diagrammatic notation used in categorical quantum mechanics to depict quantum circuits. In this notation, a triangle labelled with 0 represents a qubit state in the zero computational basis. A box labelled with H stands for a Hadamard gate. A CNOT gate is depicted with a dot connected to an $\oplus$. The other one is a controlled-Z-rotation on an angle, depicted as a box labelled with $R_\alpha(\theta_i)$ connected to a control qubit, where α can be x, y or z, and θ any angle from 0 to 2π. An upside-down triangle labelled with 0 is a measurement in the computational basis, post-selected to be zero.

The 1-layer IQP ansatz is depicted in Fig. 11. More layers can be used to increase the expressiveness of the ansatz. In our experiments, we used 3 layers of the IQP ansatz. A cap is mapped to a Bell state, while a single-legged ket vector is parametrised as a series of X, Z, and X single qubit rotations, which span the entire 1 qubit space. Any ket vector with more than one leg is initially prepared as an equal superposition of all computational basis states with Hadamard gates, followed by a series of controlled Z-rotations. The instances of 2-legged and 3-legged ket vectors are depicted in Fig. 11 as examples. A spider, or Frobenius multiplication, is mapped to a CNOT gate, supplied with an ancilla qubit in the zero state. The bra versions of these mappings are simply the adjoints, appearing as inverted versions of their respective kets (Figs. 12, 13, 14, 15, 16, 17, 18, and 19).

6 Experimentation

We train binary classification models to predict which noun phrase from the first sentence is the one the pronoun in the second sentence refers to. This training is a classical-quantum hybrid training scheme where a quantum computer is responsible for computing the meaning of the sentence by connecting the quantum states in a quantum circuit. A classical computer is used to compute the loss function of the training. In each iteration, a new set of quantum states is computed based on the loss function of the previous iteration.

6.1 Task

One common discourse structure involves a pronoun and its corresponding referent. Referents can occur before or after the pronoun, and the phenomenon is called *anaphora* or *cataphora*, respectively. Pronouns can be *definite* or *indefinite*. Definite pronouns refer to a specific person or object; examples of them are *I, you, he/she. it, us, we*. These are contrary

[2] A map that is determined by choices such as the number of qubits associated with each wire in the string diagram and the specific parameterized quantum states corresponding to each word. For our experiments, we represented the noun wires and sentence wires by one-qubit spaces.

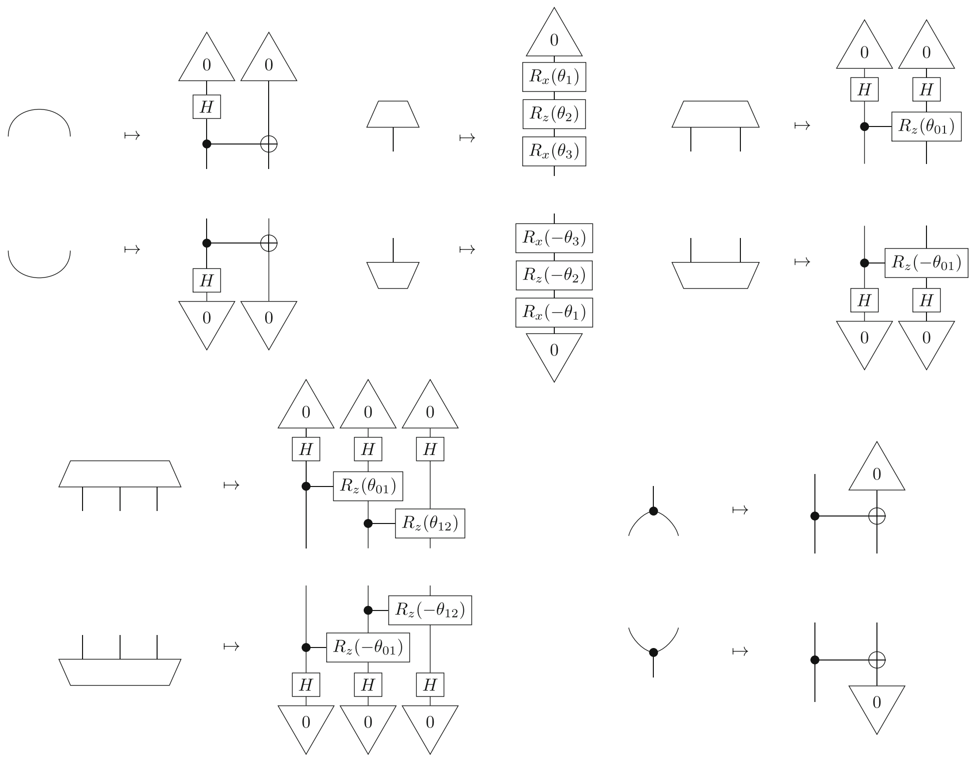

Fig. 11 Translation from string diagrams to parameterised quantum circuits using the single-layer IQP ansatz, where each grammatical type is mapped to a 1-qubit space

to *indefinite* pronouns, which do not refer to a specific person or object, e.g. *everyone* and *everything*. The overall task of finding a referent for a pronoun is called pronoun resolution, and depending on the type of the pronoun, the adjective definite or indefinite is used. Pronoun resolution is a specific case of a broader challenge known as coreference resolution (CR). This process involves identifying and grouping all text expressions, referred to as *mentions*, in a text that refer to the same entity or event. It is a critical task for achieving natural language understanding, as misidentifying or mismatching references can introduce biases and hinder accurate comprehension. CR involves three main sub-tasks: (1) mention detection, (2) determining coreference between two mentions, and (3) clustering mentions based on their coreference relationships. While all three sub-tasks are important, a significant focus of coreference resolution research lies in the second part—predicting whether two mentions are coreferent or not. In this experiment, we train models with true or false categories as explained below.

Given a discourse of two sentences S1 and S2, where S2 includes a pronoun and S1 a few noun phrases, the classifier will return 'true' if the pronoun in S2 refers to the correct candidate referent in S1, and 'false' if it does not. As examples, consider the following discourse:

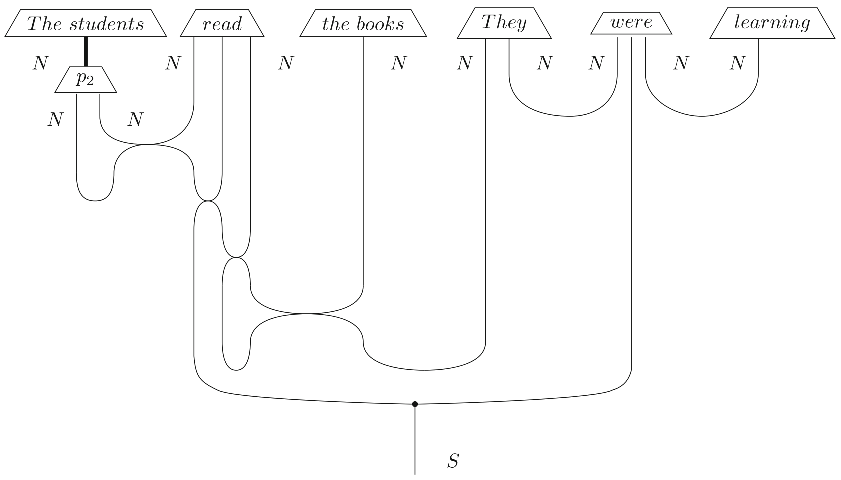

Fig. 12 Diagram representing the discourse: *'The students read the books. They were learning.'* For the sake of clarity, we treat the determiner-noun phrases *The students* and *The books* as single units. Since determiners are eventually discarded in the rewriting process, this simplification will not affect the end result. The phrase *The students* is a Fock state which is projected to the 2-copies subspace by the projection map p_2

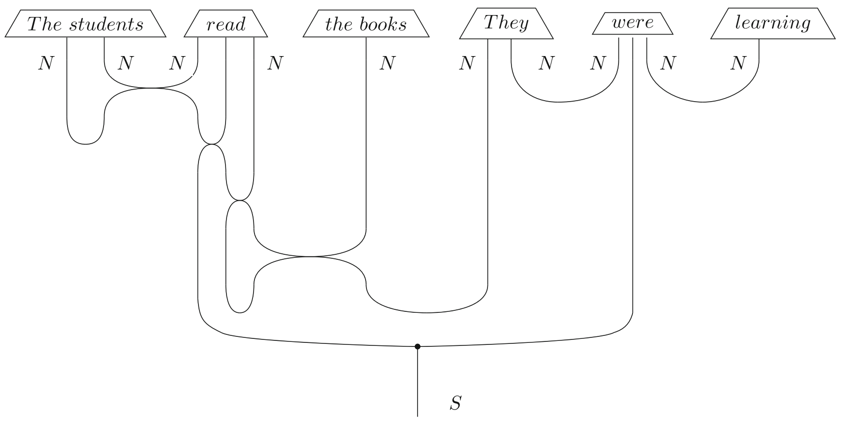

Fig. 13 Diagram after the Fock Space Projection simplification step. Now the phrase *The students* is a state in the 2-copies subspace of the Fock space

'The trophy doesn't fit into the brown suitcase.' (S1).
'It is too large.' (S2).

This discourse is labelled as **true** if the pronoun **'It'** refers to **'The trophy'** and **false** if it refers to **'the brown suitcase'**. Compare the above discourse to the following, which has the opposite classification:

'The trophy doesn't fit into the brown suitcase.' (S1).
'It is too small.' (S2).

It is labelled **true** if **'It'** refers to **'the brown suitcase'** and **false** if it refers to **'The trophy'**.

The traditional linguistic method of resolving a pronoun is by checking its binding constraints, i.e. different types of agreements between a pronoun and its candidate referents. What makes our task challenging is that its pronouns are already in all linguistic agreements with all of their candidate referents and thus binding constraints method is not useful in solving it. This task is known as the Winograd Schema Challenge and was introduced in Levesque et al. (2012) as a test of intelligence and an alternative to Turing's test. It is designed in such a way that solving it requires extra-linguistic (or world) knowledge. For instance, in the example discourse units above, this extra knowledge is that if an object does not fit into another object, the first object is probably too big and/or the second one too small. The QNLP pipeline, capable of encoding both syntactic and semantic linguistic structures, is well suited for training on this task.

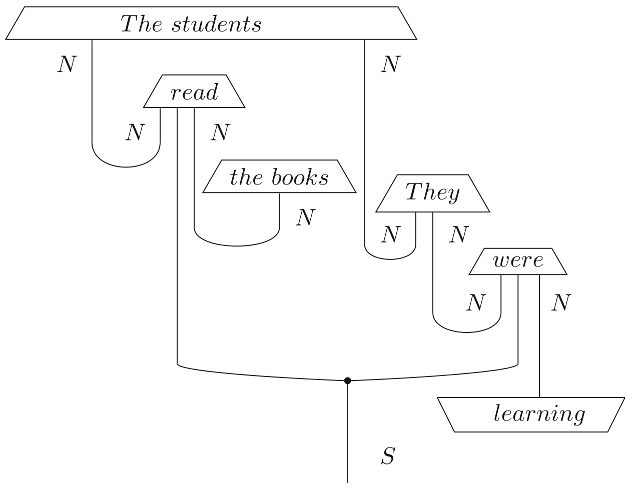

Fig. 14 Diagram is simplified by removing excessive cups and stretching wires

6.2 Dataset

The process of training variational quantum circuits (VQC) in QNLP involves the optimisation of multiple parameters associated with each word in a given dataset, with the objective of minimising the loss value on the training set. When it comes to predicting the output of a test sample, a VQC is constructed based on the input sentence. Each word in the sentence is associated with a specific set of parameters, which are learnt during the training process. However, a significant challenge arises when we encounter a word in the test set that is absent in the training set. Since this word does not have a readily available parameter assignment, we refer to this problem as the out-of-vocabulary (OOV) problem. To tackle the OOV problem, we define a fixed set of words with grammatical relations between them and prompt the GPT−3.5 (Brown et al. 2020) model to generate pairs of sentences that exhibit a substantial overlap in vocabulary.

We start by selecting a pair of sentences from the definite pronoun resolution dataset developed in Rahman and Ng (2012), which itself was generated as an extension of the Winograd Schema Challenge dataset (Levesque et al. 2012). Each pair consists of a sentence that contains two referents, followed by a subsequent sentence that incorporates a pronoun. The pronoun is carefully constructed to align with respect to gender, number, and semantic class to each of the candidate referents mentioned in the first sentence. For instance, consider the following example taken from the dataset (Rahman and Ng 2012):

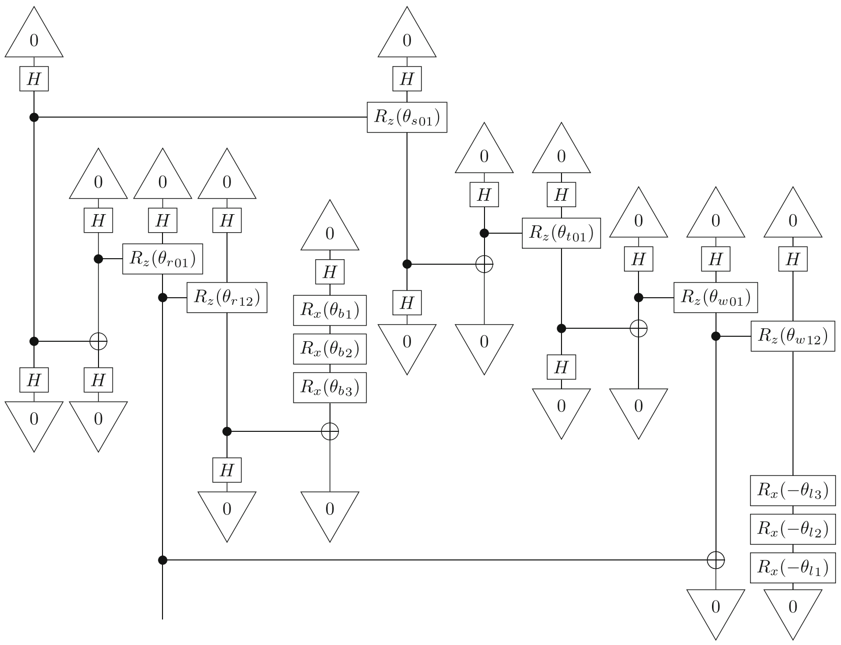
Fig. 15 The final optimised diagram is transformed into parameterised quantum circuit, using the IQP ansatz shown in Fig. 14

– *The students read the books. They were learning.*
– *The students read the books. They were interesting.*

We build on this example by expanding the range of sentence structures that can have the same vocabulary. We do so by systematically creating a variety of grammatical combinations using the following templates:

– The students *(verb, phrasal verb, verb phrase)* the books. They were *(adjective, gerund phrase)*.
– The *(adjective)* students *(verb, phrasal verb, verb phrase)* the books.
They were *(adjective, gerund phrase)*.
– The students *(verb, phrasal verb, verb phrase)* the *(adjective)* books.
They were *(adjective, gerund phrase)*.
– The *(adjective)* students *(verb, phrasal verb, verb phrase)* the *(adjective)* books.
They were *(adjective, gerund phrase)*.

Here, the verb *read* can be replaced by another *verb*, *phrasal verb*, or a *verb phrase*. Similarly, the adjectives *learning/interesting* can be replaced by another *adjective* or *gerund phrase*. We list all combinations for different examples in the Appendix section.

Next, we utilise the prompt provided below in GPT−3.5 along with the previously mentioned sentence structures. This technique is called few-shot prompting, where we provide examples in the prompt to steer the model to better performance. This enables us to generate coherent sentences while maintaining grammatical accuracy and ensuring semantic consistency. Note that the red tokens should be modified for each example.

Provide alternative sentences by replacing the words or phrases inside the brackets for each statement. Utilize different verbs, phrasal verbs, verb phrases, adjectives, or gerund phrases to create new sentences based on the given structure. Ensure that the pronoun 'they' in the second sentence refers to 'students' / Ensure that the pronoun 'they' in the second sentence refers to 'books'

Below is a sample output on the '*student*' example:

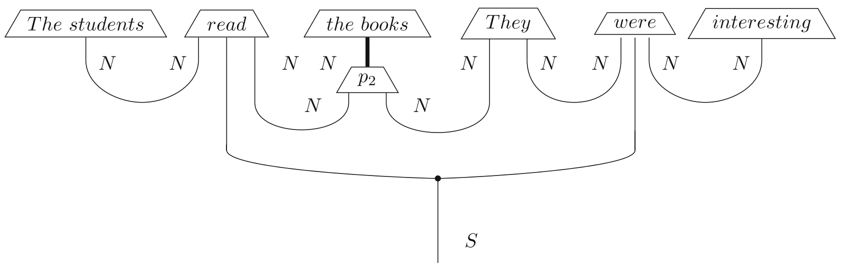

Fig. 16 Example in which the referent of the pronoun is the object: *'The students read the books. They were interesting.'*

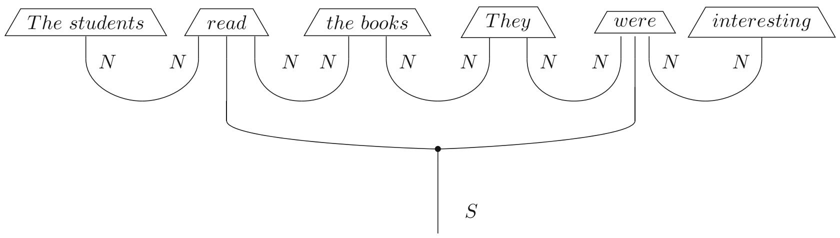

Fig. 17 Diagram after the Fock Space Projection simplification step

- The students researched the books. They were seeking new insights.
- The curious students devoured the books. They were gaining knowledge.
- The students reviewed the complex books.
  They were helping them to understand new concepts.
- T he motivated students explored the recommended books. They were containing valuable information.

This methodology provided us with a large scale dataset of 16,400 sentence pairs, comprising approximately 200,000 words, with 1214 unique words. This dataset is the largest to date in comparison to previous experiments in QNLP. The dataset was subsequently split into three subsets: 10,496 pairs ($\sim$60%) for training, 2624 pairs ($\sim$20%) for validation, and 3280 pairs ($\sim$20%) for testing. The training and testing datasets share a common vocabulary of 95%, while none of the sentence pairs in the testing set appears in the training or validation sets. This careful partitioning allows for robust evaluation and ensures that the models are tested on unseen data during the testing phase.

Due to the large size and structural complexity of the dataset, as well as limitations in conducting noisy simulations at scale, we faced challenges in running simulations on the complete dataset. As a result, we opted to sample a smaller set of 144 sentence pairs from the larger dataset. This smaller subset had a reduced vocabulary size of only 18 unique words and simpler grammatical structures. This downscaled dataset served as the basis for our noisy simulations, allowing us to conduct our analysis within the computational constraints.

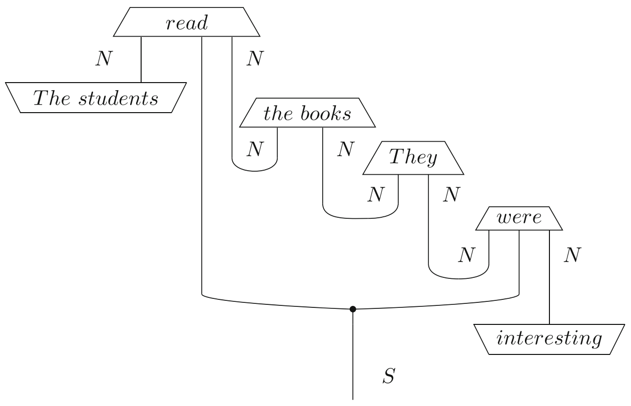

Fig. 18 Diagram is simplified by removing excessive cups and stretching wires

The code and datasets can be accessed through the following link: https://github.com/hwazni/quantumcoref.

6.3 Models

We conducted experiments using both large and small datasets. For the larger dataset, we simulated the parameterised circuits with the NumPyModel from the *Lambeq* package, implementing a noiseless non-shot-based simulation. This involved representing quantum gates as tensors and performing computations through tensor contraction. By using this simulation method, we were able to obtain the ideal probability distribution that would be achieved on a noise-free quantum computer.

For the smaller dataset, we experimented with a noisy shot-based simulation. The noisy shot-based simulation takes into account the impact of real-world quantum hardware imperfections, such as quantum gate errors, decoherence, and shot noise, thereby providing a more realistic assessment of the model's performance on an actual quantum computer. Here, we execute the circuit multiple times (8192 shots), with the outcomes averaged to approximate the ideal probability distribution. We used the TketModel from the *Lambeq* package and *AerBackend* simulator, which serves as a backend for running simulations on the *Qiskit Aer QASM* simulator. We employed a backend noise model based on the *IBMQ Lagos* device. This allows us to mimic the effects of noise, errors, and environmental interference that would be encountered on a real quantum device.

To compare the performance of our models, we also implemented and experimented with a Bag-of-Words model, where there is no grammatical structure present and the sentences of each entry are represented as multi-sets of words. Word order and any other syntactic structure is forgotten and not taken into account. Figures 20 and 21 present the Bag-of-Words string diagrams and quantum circuits of the '*students*' example.

6.4 Training

The two sentences (S_1, S_2) of each entry of the dataset are combined to create a single output quantum state. This sin-

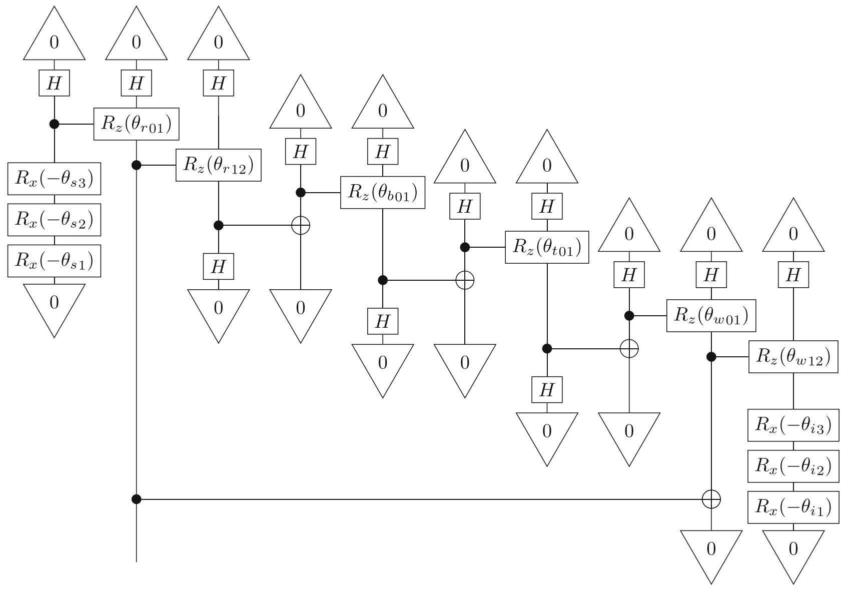
Fig. 19 The final optimised diagram of Fig. 18 is transformed into parameterised quantum circuit, using the IQP ansatz

gle state will be used as the input to our binary classifier. In principle, this can be any quantum map that takes two sentences as inputs and gives a sentence as the output. We used a CNOT gate, since it encodes a commutative Frobenius multiplication, acting in some ways similar to a logical conjunction. The resulting combined quantum circuit is denoted by $S_1 \odot S_2$.

We use a hybrid training scheme that combines classical and quantum computing to learn the parameters of this combined circuit. The procedure is as follows: First, we randomly initialise a set of parameters $\Theta = (\theta_1, \theta_2, ..., \theta_k)$, such that every parameter used in every ansatz of every word in the vocabulary is included.

Every combined circuit from the test dataset is then evaluated against Θ. That means if the same word appears in two different circuits, it is represented as the same quantum state in both circuits. We denote the output state from the circuit $S_1 \odot S_2$ as $|S_1 \odot S_2\,(\Theta)\rangle$. The expected prediction of the binary class of each entry of the dataset is then given by the Born rule, i.e. as follows:

$$l^i_\Theta(S_1 \odot S_2) := |\langle i\,|S_1 \odot S_2\,(\Theta)|\rangle|^2 + \epsilon$$

In the above, $i \in \{0, 1\}$, $\epsilon = 10^{-9}$ and $l_\Theta(S_1 \odot S_2)$ is a probability distribution defined as follows:

$$l_\Theta(S_1 \odot S_2) := (l^0_\Theta(S_1 \odot S_2), l^1_\Theta(S_1 \odot S_2))/\sum_i l^i_\Theta(S_1 \odot S_2)$$

Note that the sum $\sum_i |\langle i\,|S_1 \odot S_2\,(\Theta)|\rangle|^2$ need not equal to 1 as there can be post-selections in the circuits, rendering some circuit runs discarded. In some extreme cases, almost

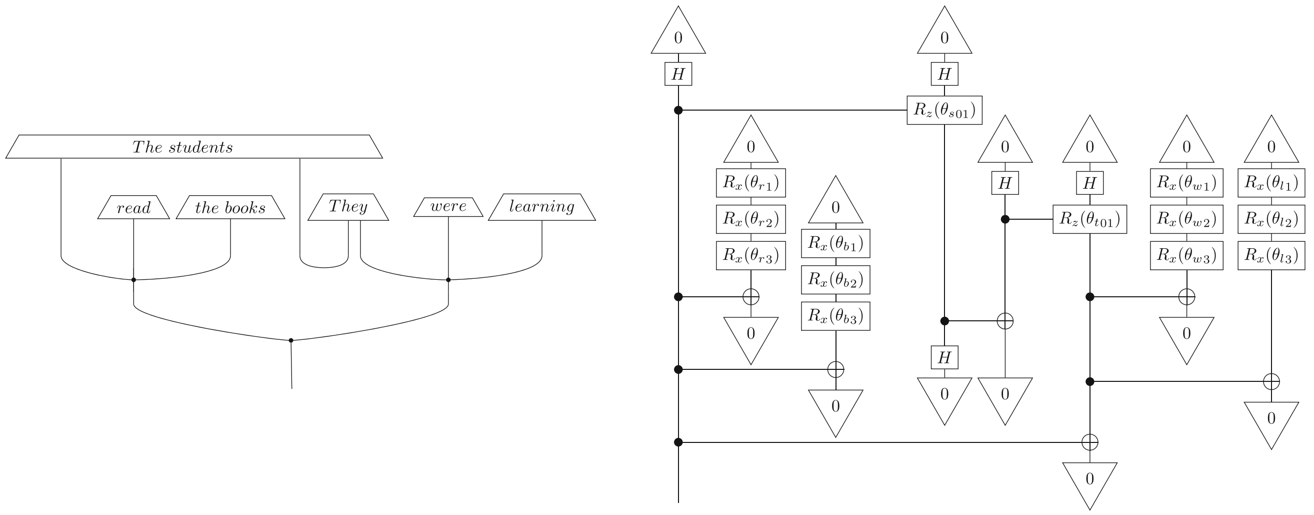

Fig. 20 A Bag-of-Words diagram representing the discourse: *'The students read the books. They were learning.'* along with its transformation into a parametrised quantum circuit

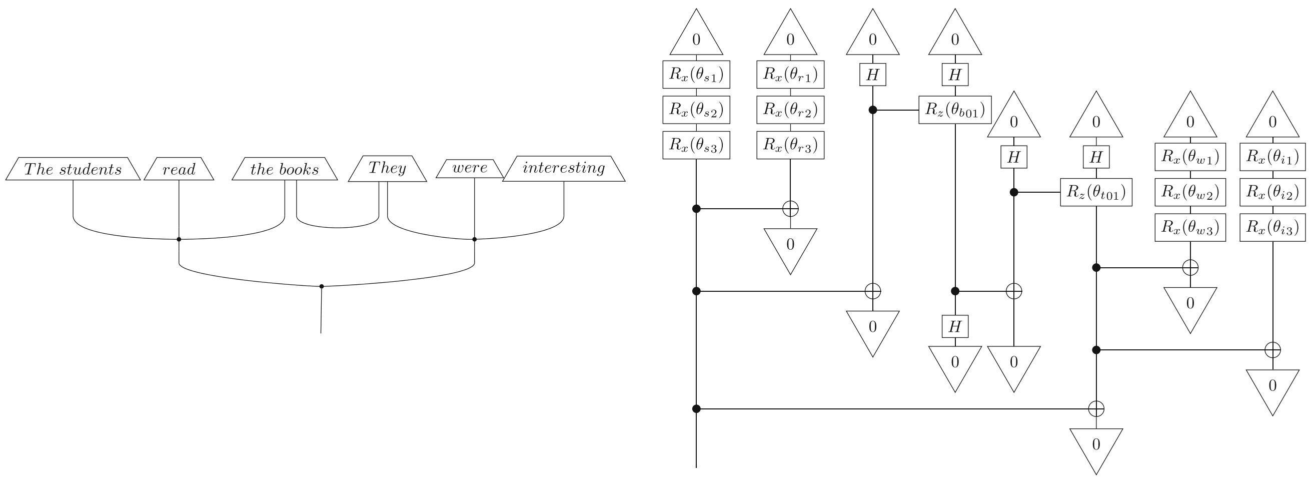

Fig. 21 A Bag-of-Words diagram representing the discourse: *'The students read the books. They were interesting.'* along with its transformation into a parametrised quantum circuit

all circuit runs could be discarded. To avoid the possibility of dividing by zero, the small constant $\epsilon = 10^{-9}$ is added. The predicted binary class from the model is then obtained by rounding to the nearest integer $\lfloor l_\Theta(S_1 \odot S_2) \rceil$.

The class labels are represented as one-hot encoding [0,1] and [1,0], corresponding to *incorrect* referent and *correct* referent, respectively. The model is trained by comparing the predicted label with the training label using a binary cross-entropy loss function and minimised using a non-gradient-based optimisation algorithm known as SPSA (Simultaneous Perturbation Stochastic Approximation) (Spall 1998). As a result, the system learns to classify the sentences by adjusting the parameters.

For the hyper-parameters, we used an initial learning rate $a = 0.05, 0.1$, an initial parameter shift scaling factor $c = 0.06$, and a stability constant A = 10, 20 and 1000, 2000 training steps.

Algorithms 1 and 2 outline the training and evaluation processes of the model. The input sentences, represented by pairs S1 and S2 along with the candidate referent R and pronoun P, undergo a transformation into SLLM diagrams, followed by optimization and conversion into quantum circuits. For each word in V_{train}, random parameter vectors are initialised and then concatenated into a single vector θ^0, which serves as the parameter initialisation for the training phase. Through multiple epochs, the optimizer refines the parameters, which are stored for later use during model evaluation.

During evaluation, the trained model is deployed on a test dataset, where the input sentences are converted into quantum circuits. Leveraging the parameter assignments obtained from the training phase, these parameters are utilised to configure the parameters in the Variational Quantum Circuits (VQCs). The resulting concrete circuits are contracted, enabling the calculation of the model's predictions. To assess the model's performance on the test dataset, the accuracy and loss are computed by comparing the model's predictions against the expected output values.

Algorithm 1 Training

```
Input: Training data $x_{train}$ and corresponding labels $y_{train}$
Output: Learned parameter values for each word in the training vocabulary
1: SLLM-diagrams ← [TEXT-TO-SLLM-DIAGRAM($x_i$) for each $x_i$ in $x_{train}$; $x : S_1, S_2, R, P$]
2: rewritten-SLLM-diagrams ← [REWRITE($d_i$) for each $d_i$ in SLLM-diagrams]
3: circuits ← [ANSATZ($d_i$) for each $d_i$ in rewritten-SLLM-diagrams]
4: for each $word_i \in V_{train}$
5:     $\vec{\theta}_i^{\,0}$ ← INITIALIZE()
6: end for
7: $\theta^0 \leftarrow \langle \vec{\theta}_1^{\,0}, \vec{\theta}_2^{\,0}, ..., \vec{\theta}_{|V_{train}|}^{\,0} \rangle$
8: for $t$ in range(1, epochs)
9:     $\theta^t \leftarrow$ SPSA-ITERATION($circuits, y_{train}, \mathcal{L}, \theta^{t-1}$)
10: end for
11: return $parameters \leftarrow \{word_i : \vec{\theta}_i^{\,epochs}$ for each $word_i \in V_{train}$ }
```

7 Results and analysis

Figures 22 and 23 illustrate the convergence of the SLLM and Bag-of-Words models over various training scenarios. In the noiseless simulations, which were done on the large-scale dataset, the SLLM model shows a consistent reduction in training loss, culminating at a final value of 0.44564 after 2000 iterations. On the other hand, the Bag-of-Words model demonstrates a slower rate of convergence, settling at 0.65976. Furthermore, the SLLM model surpasses in accu-

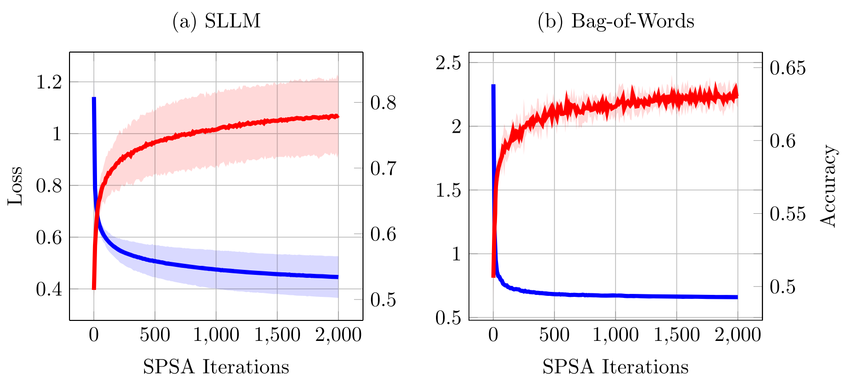

Fig. 22 Performance of noiseless simulation: average training loss and accuracy across 5 runs on the large dataset. The shaded regions represent the ± 2 standard deviation around the mean

Algorithm 2 Evaluation

```
Input: Testing data $x_{test}$, labels $y_{test}$, learned parameter values $parameters$
Output: Test accuracy and loss
1: Do steps 1, 2, 3 of Algorithm 1 with $x_{test}$
2: for each $word_i \in V_{test}$
3:     $\vec{\theta}_i \leftarrow parameters\,[word_i]$
4: end for
5: $\theta_{test} \leftarrow \langle \vec{\theta}_1, \vec{\theta}_2, ..., \vec{\theta}_{|V_{test}|} \rangle$
6: $y_{pred} \leftarrow$ PREDICT(circuits, $\theta_{test}$)
7: accuracy $\leftarrow$ MEASURE-ACCURACY($y_{test}, y_{pred}$)
8: loss $\leftarrow$ MEASURE-LOSS($y_{test}, y_{pred}, \mathcal{L}$)
9: return accuracy, loss
```

racy, achieving 0.78046 versus the Bag-of-Words model's 0.63046 at the 2000th iteration.

In the noisy simulations conducted on the small-scale dataset, the SLLM model demonstrates a gradual decrease in training loss, reaching 0.6628, while the Bag-of-Words model maintains a relatively consistent loss of approximately 0.68542. The SLLM model shows incremental improvement in accuracy, reaching 0.5597, while the Bag-of-Words model's accuracy remains stable at around 0.5.

We also did noiseless simulations on the small-scale experiments. Here, both models experienced a decrease in training loss. However, the SLLM model achieves a lower minimum loss of 0.2718 at iteration 1000, while the Bag-of-Words model plateaus at 0.12572. The SLLM model also exhibits a steady increase in accuracy, reaching 0.90556 compared to the Bag-of-Words model's accuracy of 0.95832 at the same iteration. These results highlight the superior performance of the SLLM model compared to the Bag-of-Words model in terms of both loss reduction and accuracy improvement across different training scenarios.

Table 2 presents the test accuracy results for the noiseless simulations. The SLLM model achieves an average test accuracy of 72.65%, indicating its ability to accurately classify and predict outcomes on the test dataset. On the other hand, the Bag-of-Words model achieves a lower average test accuracy of 55.99%. Across the five runs, the SLLM model exhibits higher accuracy, ranging from 67.48 to 77.23%, while the Bag-of-Words model's accuracy varies from 55.57 to 56.79%.

Table 3 displays the test accuracy results for the noisy simulations, conducted on both the SLLM and Bag-of-Words models. The SLLM model achieves an average test accuracy of 55.27%, while the Bag-of-Words model has a slightly lower average test accuracy of 50.27%. This suggests that both models struggle to perform well in the presence of noise,

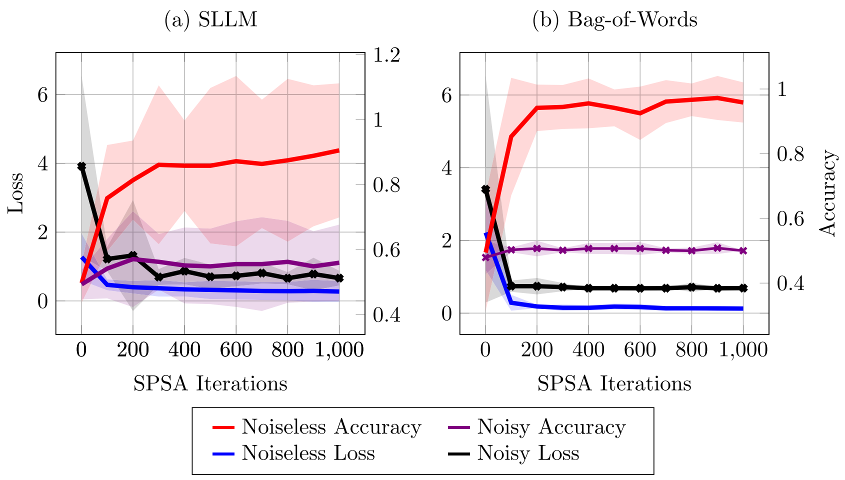

Fig. 23 Performance of noiseless and noisy simulations: average training loss and accuracy across 5 runs on the small dataset. The shaded regions represent the ± 2 standard deviation around the mean

Table 2 Noiseless test accuracy across 5 runs—the large scale experiment

Noiseless SLLM	Noiseless Bag-of-Words
75.64%	55.70%
67.48%	56.21%
72.76%	55.70%
77.23%	56.79%
70.18%	55.57%
72.65%	**55.99%**

with the SLLM model demonstrating a slightly higher accuracy.

When comparing the noisy and noiseless simulations, it is evident that the presence of noise has a detrimental effect on the performance of both models. The test accuracy decreases for both models in the noisy simulation compared to the noiseless simulation. However, the SLLM model still exhibits better performance compared to the Bag-of-Words model in both scenarios.

The analysis of the provided results highlights the influence of dataset characteristics on the performance of the SLLM and Bag-of-Words models. Recall that the large dataset consisted of a substantial number of sentence pairs (16,400) and a larger vocabulary of 1214 unique words. In contrast, the small dataset comprised a limited number of sentence pairs (144) with a smaller vocabulary of 18 unique words. The SLLM model demonstrated its effectiveness in handling the complexity of grammatical combinations and achieved superior results on the large dataset. However, the Bag-of-Words model showcased its strength in simpler linguistic patterns where little structure was present, yielding comparable performance on the small dataset.

To assess the performance of our QNLP pipeline in comparison to classical pipelines, we evaluated different pronoun resolution architectures, namely CoreNLP (Manning et al. 2014), Neural Coreference (Clark and Manning 2016, b), and SpanBERT (Lee et al. 2018). CoreNLP combines rule-based methods with statistical models to resolve coreferences. By analysing linguistic features and contextual information, it identifies and links expressions that refer to the same entity. Neural Coreference, as its name implies, utilizes neural network architectures to model coreference relationships. Through deep learning techniques, it captures patterns and dependencies in textual data, enhancing the accuracy of coreference resolution. In contrast, SpanBERT is a specialised variant of the BERT (Devlin et al. (2019)) model (Bidirectional Encoder Representations from Transformers) fine-tuned specifically for coreference resolution tasks. It employs a transformer-based architecture to learn contextual representations of words and phrases, enabling robust and effective coreference resolution.

Table 3 Noisy test accuracy across 5 runs—the small scale experiment

Noisy SLLM	Noisy Bag-of-Words
51.38%	50%
59.72%	50%
50%	50%
63.88%	50%
51.38%	51.38%
55.27%	**50.27%**

Table 4 Test accuracy of classical coreference models compared with quantum one

CoreNLP	SpanBERT	Neural Coreference	Quantum Coreference
39.3%	84.5%	41.3%	77.5%

Our own proposed method is labelled as *Quantum Coreference* in Table 4. *Quantum Coreference* is made of two sub-modules: (1) a mentions-detection module which uses SpaCy's part-of-speech parser to identify a set of potential coreference mentions, and (2) our trained SLLM hybrid quantum-classical classifier which computes a coreference score for each pair of potential mentions. The results showed varying levels of performance among these methods. CoreNLP achieved the lowest accuracy score of 39.3%, indicating limitations in accurately resolving coreferences in the given dataset. SpanBERT demonstrated the highest accuracy of 84.5%, showcasing its superior ability to capture contextual information and successfully resolve coreferences. Neural Coreference achieved a moderate accuracy of 41.3%, falling behind SpanBERT but outperforming CoreNLP. Our Quantum Coreference method showed promising results with an accuracy of 77.5%, suggesting the potential of quantum NLP techniques in coreference resolution. Wazni and Sadrzadeh (2023) merged the quantum and classical approaches and noticed further improvements.

8 Summary, conclusions, and future work

This paper had two main parts. The first part was on the theory of a specific modal Lambek calculus and its applications to modelling discourse relations such as the coreference between a pronoun and a noun phrase. In this part, we focused on the calculus of Kanovich et al. (2020), which we denote by **SLLM**. The modalities of this calculus are in the style of the *soft* modalities of Linear Logic (Lafont 2004). They have the advantage that the logic containing them remains decidable, despite allowing for an implicit form of copying—via the notion of storage and projections from it. A vector space semantics for this calculus was developed in previous work

(McPheat et al. 2020), where the modal formulae were interpreted as *truncated Fock spaces*, i.e. direct sums of tensor powers, but only up to a fixed given order k_0. We end the first part by introducing a novel string diagrammatic semantics for this truncated Fock space semantics.

In the second part, we followed the line of work initiated in Meichanetzidis et al. (2020); Lorenz et al. (2021) and translated our string diagrams to quantum circuits. Here, vector spaces are translated to quantum states and the operations on them to quantum gates. We translate our truncated Fock space semantics into quantum circuits by extending the existing translation. We then applied our setting to a definite pronoun resolution task and (1) developed a much larger version of the original dataset of Levesque et al. (2012), elaborated on in Rahman and Ng (2012), (2) modelled the coreference relations with modal formulae, (3) built the corresponding Fock space semantics of them, (4) depicted the reasoning in novel string diagrams, and (4) translated these latter to variational quantum circuits. We learnt the parameters of the resulting circuits by simulating them in the noiseless and noisy simulators of IBMQ (Qiskit 2021). In order to make accurate comparisons, we also implemented and experimented with a model containing no notion of structure in it. The highest accuracies were recorded for the structured model.

In conclusion, our study highlights the potential of quantum-inspired NLP techniques in enhancing coreference resolution. Our quantum model demonstrated superior performance compared to traditional approaches, indicating the effectiveness of quantum methods in handling linguistic complexities. However, further research and advancements are needed to narrow the gap with state-of-the-art transformer-based models like SpanBERT. Integrating quantum-inspired NLP (QNLP) with transformer architectures could unlock further improvements in coreference resolution tasks. Exploring the use of real quantum computers for training circuit parameters instead of relying solely on simulations is another avenue for future work. Finally, efforts to develop Fock space quantum computers and develop NLP packages for them are under way (Clément et al. 2022; de Felice and Coecke 2022). Employing these developments could provide more natural and powerful means for our semantics.

Appendix

- Example 1:
  - The sniper *{verb, phrasal verb}* the terrorist. He was a *{adjective} {noun, compound noun}*
  - The *{adjective}* sniper *{verb, phrasal verb}* the terrorist. He was a *{adjective} {noun, compound noun}*
  - The sniper *{verb, phrasal verb}* the *{adjective}* terrorist. He was a *{adjective} {noun, compound noun}*
  - The *{adjective}* sniper *{verb, phrasal verb}* the *{adjective}* terrorist. He was a *{adjective} {noun, compound noun}*
- Example 2:
  - The heart *{verb}* blood *{adverb, adverbial phrase}*. It was a *{adjective} {noun}*
  - The *{adjective}* heart *{verb}* blood *{adverb, adverbial phrase}*. It was a *{adjective} {noun}*
  - The heart *{verb} {adjective}* blood *{adverb, adverbial phrase}*. It was a *{adjective} {noun}*
  - The *{adjective}* heart *{verb} {adjective}* blood *{adverb, adverbial phrase}*. It was a *{adjective} {noun}*
- Example 3:
  - The people *{verb, phrasal verb, verb phrase}* against the government. They were *{gerund phrase}*
  - The *{adjective}* people *{verb, phrasal verb, verb phrase}* against the government. They were *{gerund phrase}*
  - The people *{verb, phrasal verb, verb phrase}* against the *{adjective}* government. They were *{gerund phrase}*
  - The *{adjective}* people *{verb, phrasal verb, verb phrase}* against the *{adjective}* government. They were *{gerund phrase}*
- Example 4:
  - The birds *{verb, phrasal verb, verb phrase}* the seeds. They were *{adjective, gerund phrase, prepositional phrase}*
  - The *{adjective}* birds *{verb, phrasal verb, verb phrase}* the seeds. They were *{adjective, gerund phrase, prepositional phrase}*
  - The birds *{verb, phrasal verb, verb phrase}* the *{adjective}* seeds. They were *{adjective, gerund phrase, prepositional phrase}*
  - The *{adjective}* birds *{verb, phrasal verb, verb phrase}* the *{adjective}* seeds. They were *{adjective, gerund phrase, prepositional phrase}*
- Example 5:
  - The bee *{verb, phrasal verb, verb phrase}* the flower. It was *{adjective, noun phrase, prepositional phrase, gerund phrase}*
  - The *{adjective}* bee *{verb, phrasal verb, verb phrase}* the flower. It was *{adjective, noun phrase, prepositional phrase, gerund phrase}*
  - The bee *{verb, phrasal verb, verb phrase}* the *{adjective}* flower. It was *{adjective, noun phrase, prepositional phrase, gerund phrase}*
  - The *{adjective}* bee *{verb, phrasal verb, verb phrase}* the *{adjective}* flower. It was *{adjective, noun phrase, prepositional phrase, gerund phrase}*

- Example 6:
  - The storm *{verb, verb phrase}* the flight. It was *{gerund phrase}*
  - The *{adjective}* storm *{verb, verb phrase}* the flight. It was *{gerund phrase}*
  - The storm *{verb, verb phrase}* the *{adjective}* flight. It was *{gerund phrase}*
  - The *{adjective}* storm *{verb, verb phrase}* the *{adjective}* flight. It was *{gerund phrase}*
- Example 7:
  - The sailors *{verb, phrasal verb, verb phrase}* the boats. They were *{adjective, gerund phrase}*
  - The *{adjective}* sailors *{verb, phrasal verb, verb phrase}* the boats. They were *{adjective, gerund phrase}*
  - The sailors *{verb, phrasal verb, verb phrase}* the *{adjective}* boats. They were *{adjective, gerund phrase}*
  - The *{adjective}* sailors *{verb, phrasal verb, verb phrase}* the *{adjective}* boats. They were *{adjective, gerund phrase}*
- Example 8:
  - The students *{verb, phrasal verb, verb phrase}* the books. They were *{adjective, gerund phrase}*
  - The *{adjective}* students *{verb, phrasal verb, verb phrase}* the books. They were *{adjective, gerund phrase}*
  - The students *{verb, phrasal verb, verb phrase}* the *{adjective}* books. They were *{adjective, gerund phrase}*
  - The *{adjective}* students *{verb, phrasal verb, verb phrase}* the *{adjective}* books. They were *{adjective, gerund phrase}*
- Example 9:
  - The police *{verb}* the criminals *{prepositional phrase}*. They were *{noun phrase, prepositional phrase}*
- Example 10:
  - The knife *{verb, phrasal verb}* the fence *{adverb, prepositional phrase}*. It was *{noun phrase}*
  - The *{adjective}* knife *{verb, phrasal verb}* the fence *{adverb, prepositional phrase}*. It was *{noun phrase}*
  - The knife *{verb, phrasal verb}* the *{adjective}* fence *{adverb, prepositional phrase}*. It was *{noun phrase}*
  - The *{adjective}* knife *{verb, phrasal verb}* the *{adjective}* fence *{adverb, prepositional phrase}*. It was *{noun phrase}*

Author Contributions Hadi Wazni contributed to the experimental part of the project by implementing, conducting, and analysing the experiments. Lachlan McPheat developed the theoretical part, concerning the Fock space semantics and the logic used for grammatical analysis. Kin Ian Lo provided valuable assistance in understanding quantum concepts and executing the simulations on IBMQ devices. Mehrnoosh Sadrzadeh supervised the entire project, offering guidance and insights at each stage. She provided input on the development of logic and its semantics, the creation of the large-scale dataset, and the selection of appropriate models.

Funding Mehrnoosh Sadrzadeh would like to express gratitude for the Royal Academy of Engineering Research Chair/Senior Research Fellowship RCSRF2122-14-152 on Engineered Mathematics for Modelling Typed Structures, which provided valuable funding for this research. Kin Ian Lo in would like to acknowledge the support from the Engineering and Physical Sciences Research Council under grant number EP/S021582/1. Hadi Wazni and Lachlan McPheat would like to thank the UCL CS department for the PhD scholarships that supported their work.

Availability of data and materials The code and data used in this paper are available at the following link: https://github.com/hwazni/quantumcoref. Users can access the repository to obtain the source code, datasets, and any other relevant materials used in this study.

Declarations

Conflict of interest The authors declare no competing interests.

References

Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S, Herbert-Voss A, Krueger G, Henighan T, Child R, Ramesh A, Ziegler DM, Wu J, Winter C, Hesse C, Chen M, Sigler E, Litwin M, Gray S, Chess B, Clark J, Berner C, McCandlish S, Radford A, Sutskever I, Amodei D (2020) Language models are few-shot learners

Clark S (2021) Something old, something new: grammar-based CCG parsing with transformer models

Clark K, Manning CD (2016) Deep reinforcement learning for mention-ranking coreference models. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, Texas, November 2016. Association for Computational Linguistics, pp 2256–2262

Clark K, Manning CD (2016b) Improving coreference resolution by learning entity-level distributed representations. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Berlin, Germany, August 2016. Association for Computational Linguistics, pp 643–653

Clément A, Heurtel N, Mansfield S, Perdrix S, Benoît V (2022) A graphical language for linear optical quantum circuits, Lov-calculus

Coecke B, de Felice G, Meichanetzidis K, Toumi A (2020) Foundations for near-term quantum natural language processing

de Felice G, Toumi A, Coecke B (2021) DisCoPy: monoidal categories in python. Electr Proc Theor Comput Sci 333:183–197

de Felice G, Coecke B (2022) Quantum linear optics via string diagrams

Devlin J, Chang M-W, Lee K, Toutanova K (2019) Pre-training of deep bidirectional transformers for language understanding, Bert

Havlíček V, Córcoles AD, Temme K, Harrow AW, Kandala A, Chow JM, Gambetta JM (2019) Supervised learning with quantum-enhanced feature spaces. Nat 567(7747):209–212

Humphreys JE (1972) Introduction to lie algebras and representation theory. Springer-Verlag

Jäger G (1998) A multi-modal analysis of anaphora and ellipsis. University of Pennsylvania Working Papers in Linguistics, 5(2):2

Jäger G (2006) Anaphora and type logical grammar, volume 24. Springer Science &amp; Business Media

Kanovich M, Kuznetsov S, Nigam V, Scedrov A (2020) Soft subexponentials and multiplexing. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)

Kanovich M, Kuznetsov S, Scedrov A (2016) Undecidability of the Lambek calculus with a relevant modality. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9804 LNCS:240–256

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:2110.04236

Lafont Y (2004) Soft linear logic and polynomial time. Theor Comput Sci

Lambek Joachim (1958) The mathematics of sentence structure. Am Math Mon 65(3):154

Lee Kenton, He Luheng Zettlemoyer L (2018) Higher-order coreference resolution with coarse-to-fine inference, In NAACL-HLT

Levesque HJ, Davis E, Morgenstern L (2012) The winograd schema challenge. In: Proceedings of the international workshop on temporal representation and reasoning

Lorenz R, Pearson A, Meichanetzidis K, Kartsaklis D, Coecke B (2021) QNLP in practice: running compositional models of meaning on a quantum computer

Manning C, Surdeanu M, Bauer J, Finkel J, Bethard S, McClosky D (2014) The Stanford CoreNLP natural language processing toolkit. In: Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations. Baltimore, Maryland, June 2014. Association for Computational Linguistics, pp 55–60

Ma Y, Tresp V (2021) Quantum machine learning algorithm for knowledge graphs. ACM Trans Quantum Comput 2(3)

Ma Y, Tresp V, Zhao L, Wang Y (2019) Variational quantum circuit model for knowledge graphs embedding

McPheat L, Wazni H, Sadrzadeh M (2020) Vector space semantics for Lambek calculus with soft subexponentials. In: Proceedings of the tenth international conference on Logical Aspect of Computational Linguistics

Meichanetzidis K, Gogioso S, De Felice G, Chiappori N, Toumi A, Coecke B (2020) Quantum natural language processing on near-term quantum computers

Moortgat M (1996) Multimodal linguistic inference. J Logic Lang Inform 5(3–4):349–385

Morrill G, Valentín O (2015) Computational coverage of TLG: nonlinearity. In: Proceedings of NLCS'15. Third workshop on natural language and computer science, volume 32. EasyChair Publications, pp 51–63

Morrill G, Valentín O (2016) On the logic of expansion in natural language. In: Logical aspects of computational linguistics. Celebrating 20 Years of LACL (1996–2016) 9th International Conference. LACL 2016, Nancy, France, December 5-7, 2016, Proceedings 9, Springer, pp 228–246

Qiskit A (2021) An open-source framework for quantum computing

Rahman A, Ng V (2012) Resolving complex cases of definite pronouns: the winograd schema challenge. In: Proceedings of the 2012 Joint conference on empirical methods in natural language processing and computational natural language learning. Jeju Island, Korea, jul 2012. Association for Computational Linguistics, pp 777–789

Sadrzadeh M, Coecke B, Grefenstette E (2013) Lambek vs. Lambek: functorial vector space semantics and string diagrams for Lambek calculus. Ann Pure Appl Logic 164:1079–1100

Shepherd Dan, Bremner Michael J (2009) Temporally unstructured quantum computation. Proc Royal Soc A: Math, Phys Eng Sci 2105:1413–1439

Spall JC (1998) Implementation of the simultaneous perturbation algorithm for stochastic optimization. IEEE Trans Aerosp Electron Syst 34(3):817–823

Wazni H, Sadrzadeh M (2023) Towards transparency in coreference resolution: a quantum-inspired approach

Wiebe Nathan (2019) Alex Bocharov. Matthias Troyer, and Krysta M Svore. Quantum language processing, Paul Smolensky

Yeung R, Kartsaklis D (2021) A CCG-based version of the DisCoCat framework

Zeng William, Coecke Bob (2016) Quantum algorithms for compositional natural language processing. Electr Proc Theor Comput Sci 221:67–75