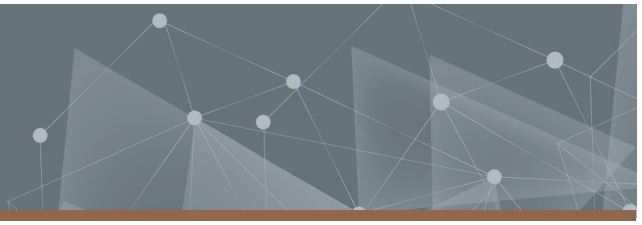
CHALMERS
UNIVERSITY OF TECHNOLOGY



Quantum Models for Word-Sense Disambiguation

Investigating the application of Compositional Distributional Models of Meaning to a word-sense disambiguation task on predicate-argument relations.

Master's thesis in Complex Adaptive Systems

Thomas Hoffmann

DEPARTMENT OF PHYSICS

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021

Quantum Models for Word-Sense Disambiguation

Investigating the application of Compositional Distributional Models of Meaning to a word-sense disambiguation task on predicate-argument relations.

Thomas Hoffmann

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2021

Quantum Models for Word-Sense Disambiguation
Investigating the application of Compositional Distributional Models of Meaning to a word-sense disambiguation task on predicate-argument relations.
Thomas Hoffmann

Supervisors:
Simon Dobnik, Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg
Mats Granath, Department of Physics, Chalmers University of Technology
David Fitzek, PhD student at the Department of Microtechnology and Nanoscience, Chalmers University of Technology

Examiner:
Mats Granath, Department of Physics, Chalmers University of Technology

Master's Thesis 2021
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Quantum Models for Word-Sense Disambiguation
Investigating the application of Compositional Distributional Models of Meaning to a word-sense disambiguation task on predicate-argument relations.
THOMAS HOFFMANN
Department of Physics
Chalmers University of Technology

Abstract

In recent years, developments in machine learning had a tremendous impact on Natural Language Processing (NLP). However, state-of-the-art language models contain billions of parameters that require vast computational resources for optimization and capture syntactic rules only from data, which does not allow an extensive analysis of the underlying logic of language. Hence, to reduce the parameter space of NLP models and close the gap between logic-based language models and statistical vector space models, Coecke, Sadrzadeh, and Clark [11] introduce a compound framework called Compositional Distributional Model of Meaning, based on Lambeks Pregroup grammar and Quantum Theory.
This thesis investigates applying the Compositional Distributional Model of Meaning on the word-sense disambiguation task by Kartsaklis, Sadrzadeh, and Pulman [18]. Different quantum embeddings are evaluated in terms of disambiguation power, given a matching context. One focus lies on the description of ambiguous words as mixed states. Mixed states are probabilistic quantum states expressed as density matrices which entail a lack of knowledge about the underlying system. Empirical data was gathered from experiments using quantum circuits and classical computations. We evaluate the performance and discuss the challenges and limitations of the current quantum computing models. The results confirm the comprehensiveness of the Compositional Distributional Model of Meaning and show statistical indications for a richer representation of words by density matrices.

Keywords: Quantum Natural Language Processing (QNLP), Compositional Distributional Model of Meaning, Word-Sense Disambiguation, Quantum Computing.

Acknowledgements

First of all, I would like to sincerely thank my supervisors Mats Granath, Simon Dobnik, and David Fitzek. Without our weekly meetings and your valuable feedback, I would have been in the dark for a long time. You took a lot of time for me, which I don't take for granted. I thank you very much for that.
A special thank you also goes to Mehrnoosh Sadrzadeh, who provided the idea for the work and joined the supervision from London. Thank you for all the meetings and the guidance!
Another personal thank you goes to my friend Leonard Salewski who supported me through many fruitful discussions!
Of course, I would also like to thank my family and all my friends. Without you, I would not be where I am now.

Thomas Hoffmann, Gothenburg, June 2021

Contents

List of Figures

List of Tables

1
Introduction

This chapter introduces the main concepts of Compositional Linguistics, Quantum Computing, and Quantum Natural Language Processing.

1.1 Introduction to Computational Linguistics

The field of computational linguistics (CL) focuses on machine-to-human interaction through natural language. As a subdomain of artificial intelligence, applications of CL range from *question answering*, *text comprehension*, *automatic translation*, to *text generation* and *speech synthesis*, among others. In recent years, many of those domains underwent a shift from statistical models to models purely based on neural networks. However, most approaches that deal with the analysis of semantics underlie the assumption that words can be embedded into vector spaces and those of similar meaning remain geometrically in close proximity [7]. As an example, such a vector space representation of meaning can be built from a text corpus where sentences are treated as *bag of words*, and meaning is derived from word co-occurrences. Establishing word vectors lets us use geometric operations like vector addition, or the inner product to reason about semantic relationships of words. One famous example is the *Queen-King* Equation:

$$\overrightarrow{\text{king}} - \overrightarrow{\text{man}} + \overrightarrow{\text{woman}} \approx \overrightarrow{\text{queen}}. \tag{1.1}$$

Word-similarities are then usually calculated through the cosine similarity [7], which is defined as

$$\text{Sim}(\overrightarrow{w_1}, \overrightarrow{w_2}) = \frac{\overrightarrow{w_1} \cdot \overrightarrow{w_2}}{\|\overrightarrow{w_1}\| \|\overrightarrow{w_2}\|}. \tag{1.2}$$

Models based on the *bag-of-words* assumption are mainly of interest for search engines or spam filters, however, only with limited applicability for more sophisticated tasks. Due to the neglected grammatical structure, sentences like “dog bites man.” and “man bites dog” are seen as equal, while the real-life implications of both sentences differ drastically.
Nowadays, the mainstream NLP focuses on inferring meaning through large neural networks. Those models are trained on vast data sets, enabling them to learn syntax and semantics from a pool of almost endless example sentences. As proven by state-of-the-art models like *BERT* [12] or *GPT-3* [6], this works very well for a broad domain of NLP tasks (question and answering, text generation, translation, etc.). However, one major drawback of those models is that the parameter space increased

tremendously to improve the accuracy (approx. 175 billion learnable parameters for GPT-3). Therefore, the training and inference of modern NLP models require a vast amount of computational resources, eventually limiting their scalability. To solve the problem, scientists seek to find more efficient and more powerful algorithms that may be realizable through Quantum Computers in the future. By nature, Quantum Computers can handle very large vector spaces, which leads to the assumption that the entire area of machine learning could benefit from quantum algorithms.

1.2 Introduction to Quantum Computing

While classical computers use bits that can be either in state 0 or 1, quantum computers use qubits, which carry physical properties that allow more powerful computational models. In most cases, qubits are physical systems that allow two distinct and measurable quantum states. A simple example would be a spin-1/2 system of an electron. Such a system has two eigenstates, namely, spin up $|\uparrow\rangle$ and spin down $|\downarrow\rangle$, which form the basis of the system's state vector. Encoding the classical bit values of 0 and 1 into the spin states of the system allows us to have a single computational entity that can be present in an infinite number of superposition of the computational basis states, defined by a state vector in a 2D complex Hilbert space.

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix} \tag{1.3}$$

with $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. Measuring the state of the system yields a 0 with probability $|\alpha|^2$ and a 1 with probability $|\beta|$. Quantum states are manipulated by quantum gates, which are unitary operators. Two examples are the NOT gate, denoted by X, which acts like a bit-flip operation, and the Hadamard-gate H, which maps the computational basis states to superpositions of equal measurement probabilities:

$$X\,|\psi\rangle = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\beta\\ \alpha\end{pmatrix} = \beta\,|0\rangle + \alpha\,|1\rangle \tag{1.4}$$

$$H\,|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \tag{1.5}$$

$$H\,|1\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) \tag{1.6}$$

To analyze the state of a 1-qubit system, we can transform the amplitudes into the form of a Bloch vector

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{\mathrm{i}\phi}\sin\frac{\theta}{2}\,|1\rangle\,. \tag{1.7}$$

Visualizing the angles in a 3D coordinate system yields the Bloch sphere:

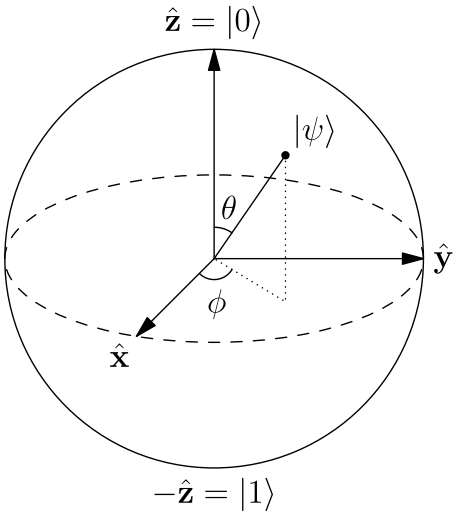


Figure 1.1: Bloch sphere used to visualize 1-qubit states[1].

Another property of quantum systems, which we leverage for quantum computing is the concept of entanglement. Two physical systems that are entangled cannot be described by two separate state vectors but one vector that lives in a compound Hilbert space. Entanglement is invoked by multi-qubit gates, like the CNOT gate:

$$\text{CNOT} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{1.8}$$

Applying the CNOT gate on two qubits defined by two unknown states $|\psi\rangle \otimes |\phi\rangle$ results in

$$\begin{aligned} \text{CNOT}\,|\psi\rangle \otimes |\phi\rangle &= \text{CNOT}\,((\alpha\,|0\rangle + \beta\,|1\rangle) \otimes (\gamma\,|0\rangle + \delta\,|1\rangle)) \\ &= \text{CNOT}\,(\alpha\gamma\,|00\rangle + \alpha\delta\,|01\rangle\,\beta\gamma\,|10\rangle + \beta\delta\,|11\rangle) \\ &= \alpha\gamma\,|00\rangle + \alpha\delta\,|01\rangle\,\beta\delta\,|10\rangle + \beta\gamma\,|11\rangle\,. \end{aligned} \tag{1.9}$$

Such a state is not separable, and a basis of the compound Hilbert space is spanned by the four vectors $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. Consequently, an entangled 3-qubit system lives in a space spanned by eight basis vectors. Hence, adding and entangling qubits to our system allows an exponential growth of the underlying Hilbert-space, which allows us to encode information into quantum state vectors that could theoretically represent the world's estimated storage capacity of approximately 2500 exabytes[2] in as few as 71 qubits (2^{71} memory slots).

[1] Graphic taken from https://de.wikipedia.org/wiki/Bloch-Kugel#/media/Datei:Bloch_Sphere.svg under CC BY-SA 3.0 licence - author: *Glosser.ca*.

[2] See statistic from 2021 here

1.3 A brief overview of Quantum NLP (QNLP)

Parallel to the field of Natural Language Processing, a second, vastly different approach of Computational Linguistics emerged within the scientific community: Quantum Natural Language Processing. Using the mathematics from categorical quantum mechanics (see [2], [9]), it was Coecke, Sadrzadeh, and Clark [11] who first introduced a unified notation of a vector spaces model of meaning and an algebraic model of language grammar based on the work of Lambek [22]. The *Categorical Compositional Distributional Model of Meaning* (CDM) enables the representation of sentence meanings within a vector space model without neglecting its grammatical structure. Identifying the similarity between the newly introduced mathematical model for language and the standard formalisms of quantum mechanics, implementing such a model on a Quantum Computer appeared to be well-suited. Hence, ten years after introducing the formalism, Meichanetzidis et al. [24] could implement the first instance of a QNLP problem on a NISQ device.

1.3.1 Language model using Pregroup grammar

Before focusing on implementing a QNLP model on a NISQ device, we have to clarify some mathematical concepts of natural language grammar rules developed by Lambek [22], [21]. The basic idea is to classify words according to their grammatical meaninig into types of mathematical categories wich undergo transformations through morphisms (see [3] for more details about category theory). According to [11], we define some building blocks of very simple English sentences as the following categories:

n = subject/noun (e.g “Mary”, “John”, ...),
j = infinitive of a verb (e.g. “likes”),
s = declarative sentence (e.g. “John likes Mary”).

Coecke, Sadrzadeh, and Clark [11] take on the definitions of Lambek [22] and state a *partially ordered monoid* $(P, \leq, \cdot, 1)$ as a category, where P is a partially ordered set of objects p that are equipped with a (non-commutative) monoid multiplication $(- \cdot -)$ with unit 1. Theoretically, we could model word types with standalone meaning, as for example nouns, in such a category. However, other word types, like verbs, need non-commutative connectors to give meaning to a sentence. Take, for example, the transitive verb “like”. To give meaning to a sentence including “like”, we must state two adjoint nouns to eventually define *who likes who*. Thus, [11] summarize word types in so-called *Pregroups* $\left(P, \leq, \cdot, 1, (-)^l, (-)^r\right)$, where each $p \in P$ has a left adjoint p^l and a right adjoint p^r, which act as the right and left inverse of p. To distinguish between the left and the right adjoint, we modify the inverse such that $p^l p \leq 1$ and $pp^r \leq 1$, which makes the operation non-commutative. Otherwise, p^l and p^r would be indistinguishable (see [22]) as

$$p^l = p^l 1 = p^l p p^r = 1 p^r = p^r \tag{1.10}$$

and we could not encode word orders into the framework. In other words, the equality of Equation 1.10 would mean that we could not model any difference between sentences like “*her favorite new restaurant*” and “*her new favorite restaurant*”.

Taking the example sentence from [11], we denote the word categories as

$$\begin{array}{ccc} \text{John} & \text{likes} & \text{Mary} \\ n & (n^r s n^l) & n \end{array}$$

As we see, the grammar reduces to a sentence type object, verifying the grammatical validity of the word combination:

$$n\left(n^r s n^l\right) n \to 1 s n^l n \to 1 s 1 \to s.$$

Note that the last equation does not show strict equality, rather we read $x \to y$ as *x is also of type y* [22]. Furthermore, the operations can be carried out step-wise, which allows us to parse the sentence into a binary tree that represents the syntactic structure of a sentence.
More complex examples can be considered by defining other word types like the adjective type a [21] or the *glueing type* σ [11], and also considering negations ("John does not like Mary"). However, we omit more complex examples and reference a more detailed description of English Pregroup grammar rules by Lambek [21].

1.3.2 Of monoidal categories and diagrams

As previously stated, Pregroups are monoidal categories that are non-commutative and compact closed. A Compact closed category $\mathbf{C_F}$ requires some properties that are described in detail by Coecke, Sadrzadeh, and Clark [11] shall be quickly summarized here:

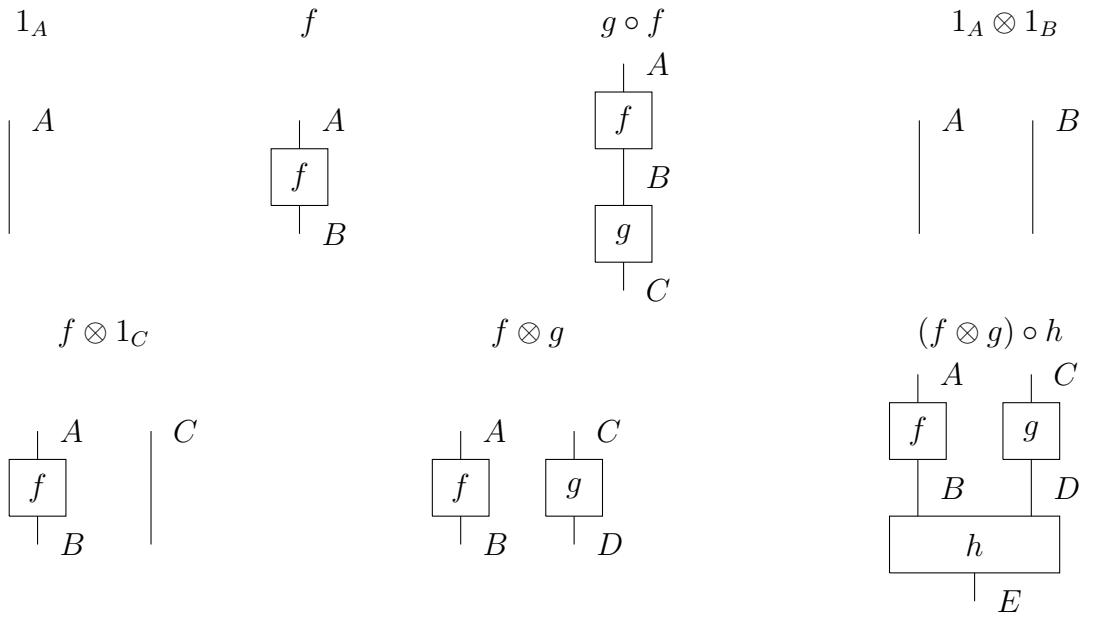
- Between each ordered pair of objects A, B of a Pregroup, a morphisms $f \in \mathbf{C_F}(A, B)$ is denoted as $f : A \to B$.
- Two morphisms $f : A \to B$ and $g : B \to C$ are concatenated by $g \circ f : A \to C$. Furthermore, morphisms are associative, i.e.

$$(h \circ g) \circ f = h \circ (g \circ f).$$

- For a pair (A, B) of ordered objects a composite object $A \otimes B$ exists. The operator $\otimes$ denotes a tensor product and is also associative.
- For each ordered pair of morphisms $(f : A \to C, g : B \to D)$ there exists a *parallel composite* $f \otimes g : A \otimes B \to C \otimes D$ that also fulfills

$$(g_1 \otimes g_2) \circ (f_1 \otimes f_2) = (g_1 \circ f_1) \otimes (g_2 \circ f_2)\,.$$

Hence, the duality of Pregroups and compact closed monoidal categories allows us to determine a vector space representation of Lambeks Pregroup grammar. From a more physical perspective, one can say that objects $A, B, C, \ldots$ of a Pregroup have the same properties as (physical) *systems* and morphisms act like *processes*. With these analogies in mind, [11] denote that these formalisms share a large overlay with systems and processes in quantum mechanics and can thus be visualized similarly. By using the diagrammatic language introduced by Coecke and Kissinger [9], we refine the operations mentioned above as



Graphics inspired from [11].

Furthermore, we define *sources* and *sinks* of our systems, which correspond to initialization and measurements of states. Sources can be expressed as *elements* of a system A and are essentially morphisms that produce the desired state from the trivial system I: $\psi : \mathrm{I} \to A$. Correspondingly, a sink takes a system A and returns the trivial system I: $\pi : A \to \mathrm{I}$ [11]. We depict both morphisms as

In the bra-ket notation a source is denoted as a *ket* $|\psi\rangle$ and a sink is denoted as a *bra* $\langle\phi|$.

1.3.3 Graphical representation of Pregroup grammar

With the symbols mentioned above, we can represent types of words as symbols consistent with the Pregroup grammar. However, we still cannot represent the connectivity of the words and how they are *glued* together. Thus, we make use of four morphisms of a compact closed category, which are

$$\eta^l : \mathrm{I} \to A \otimes A^l \quad \epsilon^l : A^l \otimes A \to \mathrm{I} \quad \eta^r : \mathrm{I} \to A^r \otimes A \quad \epsilon^r : A \otimes A^r \to \mathrm{I}. \tag{1.11}$$

These morphisms define the connection between an object A and its right and left adjoints A^r and A^l. Visually these operations are represented by *cups* ($\smile$) and *caps* ($\frown$) [9]:

Without going much more into detail here, these caps and cups can be stacked and yield the *triangle identity* shown in Figure 1.2:

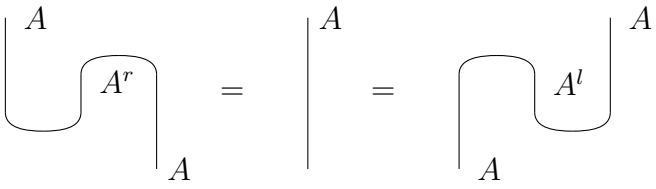


Figure 1.2: Triangle identity as described in [11], also called *yanking equation*.

The identity shown in Figure 1.2 can also be expressed as

$$(1_A \otimes \epsilon^r) \circ (\eta^r \otimes 1_A) = 1_A = (1_a \otimes \eta^l) \circ (\eta^l \otimes 1_A). \tag{1.12}$$

1.3.4 From Pregroups to natural language

Using the previously defined concepts of Pregroups as compact closed categories, Coecke, Sadrzadeh, and Clark [11] argue that it is possible to combine word meanings and grammar into one unified framework describing natural language. They first state, that word meanings can be embeded into the compact-closed vector space category **FVect** [5], where morphisms are linear maps and a tensor product between different vector spaces exists. Secondly, they denote the Pregroup grammar category introduced by Lambek [22] [21] as P. A connected product category $\mathbf{FVect} \times P$ then combines both word meanings and grammar rules in one single object. More specifically, they call an object (W, q) of $\mathbf{FVect} \times P$ a meaning space [11], containing a vector space W of word meanings, and p, the grammatical type of the word (n: noun, $n^r s n^l$: transitive verb, ...). A sentence is then the tensor product of vectors $\overrightarrow{w_i} \in W$, whose meaning is then defined by the following linear map [11]:

$$\overrightarrow{w_1 \cdots w_n} := f\left(\overrightarrow{w_1} \otimes \cdots \otimes \overrightarrow{w_n}\right) \tag{1.13}$$

We can then use the axioms of the Pregroup grammer as described in Section 1.3.1 and the mappings described in Section 1.3.2 to restore a sentence type object (S, s) within the category $\mathbf{FVect} \times P$, encoding the sentence meaning of a random grammatically valid string of words.

Example from [11] The previous sentence "John likes Mary" is an example of a positive transitive sentence with a transitive verb of Pregroup type $n^r s n^l$. The nouns are objects of the $\mathbf{FVect} \times P$, thus, we can visualize the flow of tensors within the sentence according to the schematics presented in Section 1.3.2 and 1.3.3 as in Figure 1.3.

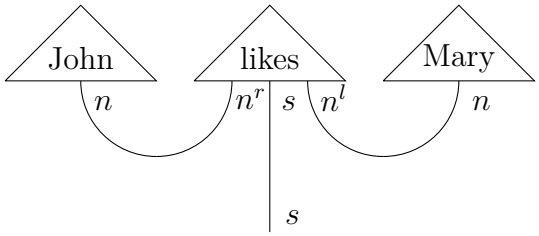


Figure 1.3: Visualization of the flow of tensors of the transitive sentence "John likes Mary".

1.3.4.1 Implementation on a quantum device

With the pregroup formalisation of language, Zeng and Coecke [37] argue that language is *quantum-native*, i.e. inherently well-suited for implementations on quantum devices. They base their assumption on the argument that the tensor product required to compute the sentence space equals simply juxtaposing qubit registers. However, the mapping of a graph as shown in Figure 1.3 to a quantum circuit is not entirely trivial. Nevertheless, in 2020, Meichanetzidis et al. [24] could implement the first QNLP model on a NISQ device. Thus, the steps to take the Pregroup grammar of a sentence and compile a quantum circuit as described in [24] shall be summarized here briefly.

General workflow First, we start with a text corpus K and classify each word within a sentence according to their Pregoup type. This procedure is based on a so-called *Part-of-Speech-Tagging* (PoS-Tagging), where we determine the word types, like transitive verbs, and assign a Pregroup type to each PoS label. This can be automated by modern neural networks, for example [4], [17]. Meichanetzidis et al. [24] then propose the following workflow:

$$\sigma \in K \xmapsto{\text{parser}} D \in \mathbf{G} \xmapsto{\text{simplify}} D' \xmapsto{\text{ansatz}} \text{QCirc} \xmapsto{\text{compiler}} \text{(NIS)QDev} \tag{1.14}$$

Essentially, each sentence $\sigma \in K$ is parsed into a diagram based on the pregroup grammar. After that, the diagram is simplified to reduce overlapping "wires" and facilitate the computation. Furthermore, each word must be translated into a quantum system, which is achieved by mapping each word type to a distinct parametrized quantum variational circuit. Last but not least, the circuit has to be compiled to run on a NISQ device. Since we have already covered parsing a sentence into a graph in the last sections, we continue with simplifying the graph.

Simplification Meichanetzidis et al. [24] introduce two different approaches to simplify the underlying graph networks. First, they elaborate on the idea introduced by Zeng and Coecke [37] which morphs a grammatical sentence diagram to a *bipartite graph*. Taking the example sentence from [24], we transform the diagram shown in Figure 1.4 into the bipartite graph shown in Figure 1.5.

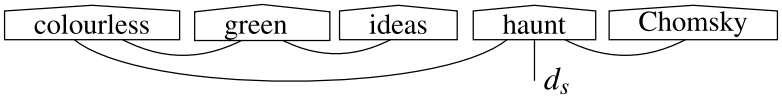


Figure 1.4: Graph parsing of the example sentence "colourless green ideas haunt Chomsky". Graphic taken from [24] under CC BY 3.0 licence.

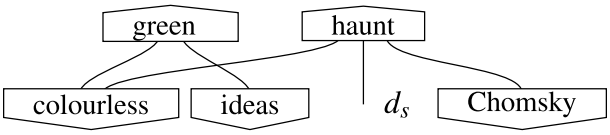


Figure 1.5: Bipartite graph of the example sentence "colourless green ideas haunt Chomsky". Graphic taken from [24] under CC BY 3.0 licence. Note that by transforming some sources into sinks, we reduce the number of quantum systems within the graph and therefore simplify the quantum circuit.

Note that in the graphics of this section, the symbols for sources and sinks introduced in Section 1.3.2 are depicted by pentagons (⬠) instead of triangles.
The underlying operation for a mapping of words (upwards pentagons ⬠) to effects (downwards pentagons ⬠) is the transpose operation on the word ansätze. Although this approach reduces the required computational resources (number of qubits and operations), [24] argue that there are also two major drawbacks

- Bipartite graphs often contain wire crossings that result in noisy swap operations in quantum circuits if we model each wire as a qubit system.
- The transpose of a word ansatz must be computed efficiently.

The first problem can be addressed by simply reordering the words. However, Meichanetzidis et al. [24] state that finding the corresponding bipartite graph with minimum wire crossings is an NP-complete problem [13]. The second problem can be resolved by requiring the word ansätze to be easily transposable.
The second proposed method is the so-called *snake removal* method. It relies on simplification properties of snakes and boxes, as for example the triangle identity shown in Figure 1.2. Referencing [24], it can be shown that the diagram of the sentence "code that runs returns results" (Figure 1.6) can be transformed into the circuit shown in Figure 1.7.

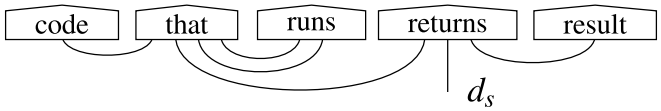


Figure 1.6: Graphic taken from [24] under CC BY 3.0 licence.

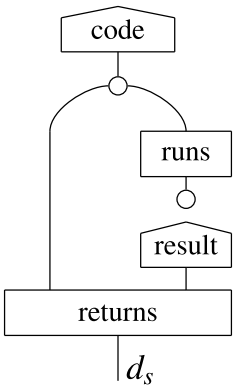


Figure 1.7: Computational graph using the representation of ZX-Calculus. Graphic taken from [24] under CC BY 3.0 licence.

Parsing To convert the simplified graphs into circuits, a suitable representation of *boxes*, *snakes*, *cups*, and *caps* has to be established. Meichanetzidis et al. [24] distinguish between the graphs from the `bigraph` (see Figure 1.5) and graphs from the `snake removal` method (Figure 1.7). To represent graphs from the `bigraph` method in quantum circuits, the following mappings are used:

- Word states (⌓) are initialzed as quantum states on a qubit register that is prepared in the $|0\rangle^{\otimes n}$ state and undergoes a unitary (parameterized) transformation.
- Word effects (⌔) are set to be the transpose of word states.
- Wire crossings are swap operations.
- Caps (⌒) are denoted as preparations of a Bell state.
- Cups (⌣) are mapped to post-selection on the Bell effect [24]

Important to mention is that variational quantum circuits are used as word ansätze. Given examples are the CNOT+U(3) circuit or a strongly entangling circuit by Schuld et al. [28], which both contain learnable parameters and are easily transposable [24]. Furthermore, one chooses the same ansatz for different words of the same grammar type.

Training In order to make use of the proposed model, the parameters of the word ansätze have to be trained on a given text corpus. Hence, Meichanetzidis et al. [24] define a toy corpus of 4 words, including two nouns. Given the word types, it is then possible to generate all sentences that are grammatically correct, i.e., whose compound word types reduce to the sentence type s. These sentences are then labeled as *True* or *False* and split into a training and validation set. Now we run the pipeline proposed in 1.14 multiple times, where the proportion of the fulfilled post-selections correspond to the *True/False* meaning. Defining a mean squared loss and using an optimization method suited for noisy functions, it is possible to find the parameters θ for the word ansätze to represent proper word meanings. A notebook of the experiment is provided by the `DisCoPy` package on Github and can be found at [16].

2
Quantum Disambiguation

In contrast to the current mainstream Transformer-based language models, Coecke, Sadrzadeh, and Clark [11] model the grammar of language as interactions of different quantum systems. Using axioms of Category Theory, they combine the mathematics of natural language grammar and the vector space model for word meanings into one compound framework. This so-called *Compositional Distributional Model of Meaning* marks a whole new approach to model sentence meanings efficiently. First realizations of QNLP problems are implemented within the `DisCoPy` framework [16], [24], [23] and show a *proof of work* of the theory. Another possible application, the disambiguation of verbs with multiple meanings, is presented in the following sections.

2.1 Verb ambiguity

In most languages, there are many words that have different meanings depending on the context where they appear in. One of many examples within the English language is the verb *to file*. Consider the example expressions

"file application" and "file nail".

In the first example, the verb takes on the meaning of *to register*, and in the second example, it has a similar meaning as *to smoothen* or *to polish*. The associated noun within the phrase disambiguates the verb and lets it collapse into the desired unique meaning. According to Clark [7], it is possible to learn a matrix representation for each verb (V) and noun (N) such that the product of a verb and a noun yields the sentence meaning (S):

$$VN = S \tag{2.1}$$

However, this approach treats the verbs *to file*, *to register* and *to polish* as three different entities, whereas it may be difficult to determine the connection between *file* and its two senses. This problem becomes more apparent if the vector representations of the two senses are far apart. The ambiguous verb would then still share some similarity to its senses, however, it may be low and the similarities to other random words could be higher. To tackle this problem, we assume that a quantum representation of ambiguous verbs may be beneficial.

2.2 Ambiguity modelled by superposition

One way of encoding ambiguous verbs into quantum states is to exploit superposition. If one encodes the unambiguous meanings of a verb into the eigenstates of a quantum system, one can define the quantum state of the verb as a superposition of those states. Using the example of *to file*, we get

$$|\text{file}\rangle = \alpha\,|\text{register}\rangle + \beta\,|\text{smooth}\rangle \tag{2.2}$$

with

$$|\alpha|^2 + |\beta|^2 = 1.$$

A quantum representation of a noun could then serve as a disambiguation entity that forces the verb to collapse into its unique meaning and yield a plausibility score for the verb-noun combination. A plausible combination would be the sentence "file nail.":

$$\langle \text{file}|\text{nail}\rangle = 1 \tag{2.3}$$

while "file person." is implausible

$$\langle \text{file}|\text{person}\rangle = 0. \tag{2.4}$$

Using the diagrammatic approach from Section 1.3.4, we can parse each verb-noun-pair into a graph as shown in Figure 2.1.

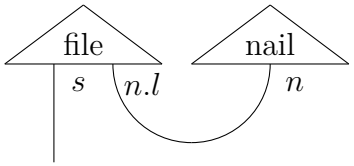


Figure 2.1: Graph parsing for "file nail".

Given a dataset of four ambiguous verbs, their single unambiguous meanings (two for each verb), and 35 nouns, Kartsaklis, Sadrzadeh, and Pulman [18] generate a corpus of 108 phrases. An extract of the dataset is shown in Table 2.1, while the full dataset is presented in Appendix A.1.

Verb	Meaning	Example Phrase
File	Register	file account file application file complaint ...
File	Smooth	file nail file teeth ...
Dribble	Drip	dribble wine dribble blood ...
Dribble	Carry	dribble ball ...
Tap	Knock	tap table ...
Tap	Intercept	tap phone ...
Charge	Bill	charge rate charge price charge rent ...
Charge	Accuse	charge defendant charge demonstrator ...

Table 2.1: Disambiguation dataset overview, example sentences, originally suggested by Kartsaklis, Sadrzadeh, and Pulman [18].

Since the dataset is composed of 4 ambiguous verbs which span a meaning space of 8 dimensions, we encode each verb-meaning (second column of Table 2.1) into the eigenstate of a quantum system:

$$\begin{aligned} |\text{file}\rangle = \alpha\,|\text{register}\rangle + \beta\,|\text{smooth}\rangle + \\ \gamma\,|\text{drip}\rangle + \delta\,|\text{carry}\rangle + \\ \epsilon\,|\text{knock}\rangle + \zeta\,|\text{intercept}\rangle + \\ \eta\,|\text{bill}\rangle + \theta\,|\text{accuse}\rangle \end{aligned} \tag{2.5}$$

where

$$|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 + |\epsilon|^2 + |\zeta|^2 + |\eta|^2 + |\theta|^2 = 1.$$

Note that we have to know the senses of the ambiguous verbs here. Using three qubits, we can encode the meaning space into the computational basis states:

$$\begin{aligned} |\text{knock}\rangle &= |000\rangle \\ |\text{carry}\rangle &= |001\rangle \\ &\dots \\ |\text{bill}\rangle &= |111\rangle \end{aligned} \tag{2.6}$$

We then investigate if it is possible to learn a quantum representation for the verbs and nouns, such that the amplitudes of each verb state correspond to the correct senses.

2.2.1 Word ansätze and circuit parsing

To map the ambiguous verbs and the set of nouns to quantum states, we have to define parameterized circuits, so-called *ansätze* [24]. For simplicity, we start with only one qubit and encode the ambiguous verb "file". In our Example, "file" can have two meanings: "register" and "smooth". We then define the meaning space of "to file" as

$$|\text{register}\rangle := |0\rangle \qquad \text{and} \qquad |\text{smooth}\rangle := |1\rangle.$$

Consequentially, the quantum state $|\text{file}\rangle$ is modelled as the following superposition:

$$|\text{file}\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,. \tag{2.7}$$

where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. Since we don't know the parameters α and β, we seek to define a parameterized ansatz for "file", such that every possible superposition can be reached. One such possibility is to initialize a qubit in the $|0\rangle$ state and apply two parameterized rotations on it, one around the x-axis and one around the y-axis:

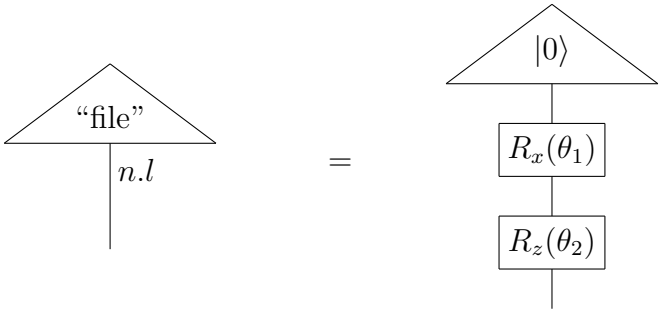


Figure 2.2: Verb ansatz for 1-qubit example.

Furthermore, we define a noun ansatz by initializing a qubit in the $|0\rangle$ state and applying a single rotation around the x-axis as shown in Figure 2.3

Figure 2.3: Noun ansatz for 1-qubit example.

Modeling the connection between the two words (i.e. the *cup*) as post-selections on the Bell effect, we get the circuit shown in Figure 2.4.

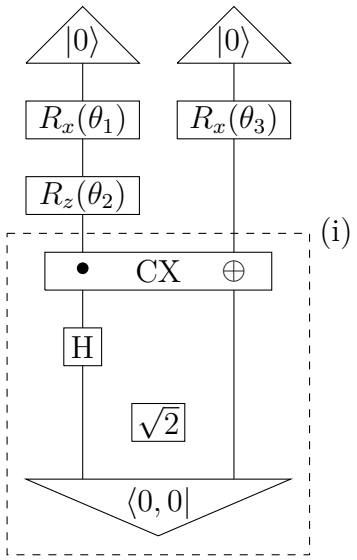


Figure 2.4: Quantum circuit for phrase "file nail.". The circuit framed by the dashed line (i) represents a *cup* and performs the post-selection on the bell-effect.

We run the circuit multiple times for each sentence and record the proportion of fulfilled post-selection conditions, which is set to be the predicted plausibility score for the given phrase. Clearly, the prediction is defined by the parameters θ, which we seek to optimize in order to model the dataset with high accuracy.

2.2.2 Parameter search

All plausible sentences are defined in the dataset given in the Appendix A.1. Extracting the entries associated with the ambiguous verb "to file", we are left with ten nouns, i. e. 30 verb-noun pairs. Each noun disambiguates the meaning of "file" into either "register" or "smooth", which is indicated by the plausibility score (see Table 2.2).

Phrase	Plausibility
file account.	1
register application.	1
smooth complaint.	0
file nail.	1
register nail.	0
smooth nail.	1
...	...

Table 2.2: Extract of the disambiguation dataset for the verb "to file".

Parsing each sentence into a quantum circuit as shown in Figure 2.4 makes it possible to calculate the plausibility scores. To compare the computed plausibilities p_i to the corresponding true values, we consider the mean squared error (MSE):

$$\mathrm{MSE}(\theta_1, \ldots, \theta_m) = \frac{1}{N} \sum_{i=1}^{N} (p_i(\theta_i) - p_{i,\mathrm{true}})^2 \tag{2.8}$$

Minimizing the loss function then yields a parameter set $\boldsymbol{\theta}$ that stores the properties of the dataset and enables us to examine the quantum state produced by the verb ansatz for "to file". Alternatively, due to the binary scores present in the dataset, one could also experiment with a cross-entropy loss. However, in more realistic scenarios, disambiguation follows rather the regression paradigm instead of binary classification; hence, the MSE seems to be the optimal choice.

2.2.3 Multi-qubit word ansätze

So far, we have only covered one-qubit ansätze, which is sufficient to represent one ambiguous verb with two distinct meanings. However, to encode the meaning space of the whole dataset from Table 2.1, we need at least three qubits. The dataset consists of 4 ambiguous verbs with 2 senses each. Thus, the meaning space is 8-dimensional, which can be encoded in the computational basis states of a 3-qubit system ($2^3 = 8$). As described in Equation 2.6, we then encode the meanings as follows:

$$\begin{array}{lcllcl} |\mathrm{knock}\rangle & = & |000\rangle & |\mathrm{accuse}\rangle & = & |100\rangle \\ |\mathrm{carry}\rangle & = & |001\rangle & |\mathrm{intercept}\rangle & = & |101\rangle \\ |\mathrm{drip}\rangle & = & |010\rangle & |\mathrm{register}\rangle & = & |110\rangle \\ |\mathrm{smooth}\rangle & = & |011\rangle & |\mathrm{bill}\rangle & = & |111\rangle \end{array} \tag{2.9}$$

Consequentially, the ansätze for the nouns and the ambiguous verbs must also consist of 3-qubit systems. Hence, we depict the graph representation of a random verb-noun phrase as shown in Figure 2.5.

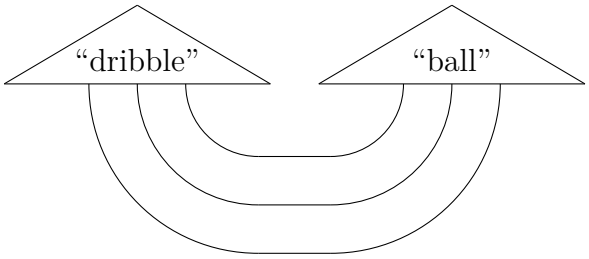


Figure 2.5: 3-qubit graph for phrase “dribble ball”.

Compiling the graph shown in Figure 2.5 as is into a quantum circuit would require 3 cups, i.e. the post-selection on 6 qubits. This is a rather resource-consuming process, therefore, we use the diagrammatic simplification [9] [24] shown in Figure 2.6.

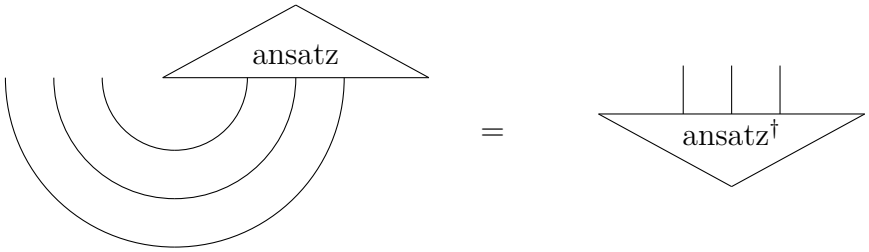


Figure 2.6: Effect ansatz (left-hand side) equals the complex conjugate of the original word ansatz (right-hand side).

Hence, the diagram shown in Figure 2.5 reduces to the simpler version in Figure 2.7 which only uses 3 qubits and projects the verb on the effect of the noun.

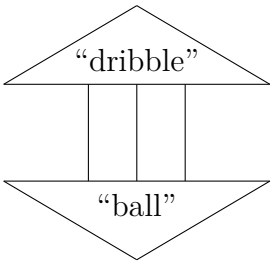
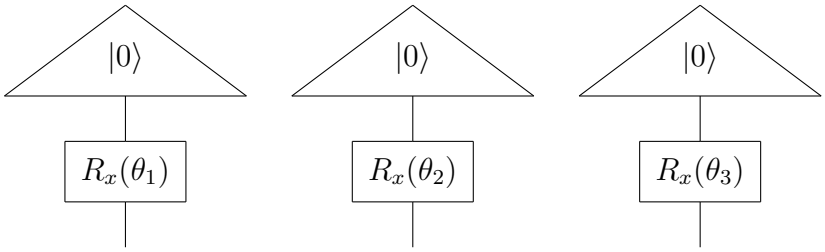


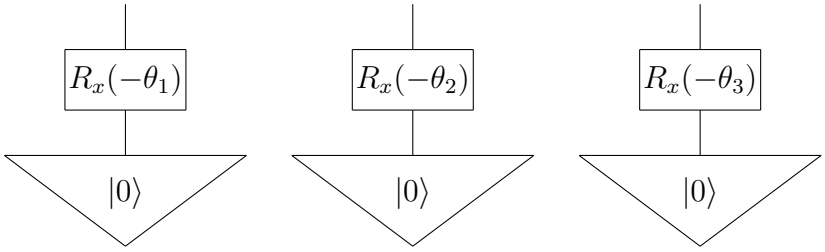
Figure 2.7: Simplified 3-qubit graph for phrase “dribble ball”.

2.2.3.1 3-qubit ansätze

To translate the graphs into quantum circuits, we have to define type-specific ansätze. We can initialize unambiguous verbs simply by applying Pauli-X gates on the $|000\rangle$ state. For nouns, we decide to apply parameterized X rotations to each qubit (see Figure 2.8a). We further see that the complex conjugate of this ansatz can be achieved by reversing the rotations and post-selecting on the $\langle 000|$ effect (see Figure 2.8b).



(a) Noun ansatz for 3-qubit model.



(b) Complex conjugate of the noun ansatz for 3-qubit model.

Figure 2.8: Noun ansatz for 3-qubit model and its complex conjugate.

To model the ansatz for ambiguous verbs, we have to propose a more complex circuit. It becomes clear when looking at the verb "tap", which consists of the single meanings "intercept" and "knock". Thus, we model the quantum representation as

$$|\text{tap}\rangle = \alpha\,|\text{knock}\rangle + \beta\,|\text{intercept}\rangle = \alpha\,|000\rangle + \beta\,|101\rangle\,. \tag{2.10}$$

For the non-trivial case of $0 < \alpha^2, \beta^2 < 1$ we have an entangled state that cannot be initialized only by using single-qubit rotations. The state preparation requires at least one entangling unit, like the CNOT-gate. Hence, Meichanetzidis et al. [24] suggest a family of parameterized and versatile state preparation circuits called the CNOT+U(3) ansätze, which are depicted in Figure 2.9 in ZX-calculus style [8].

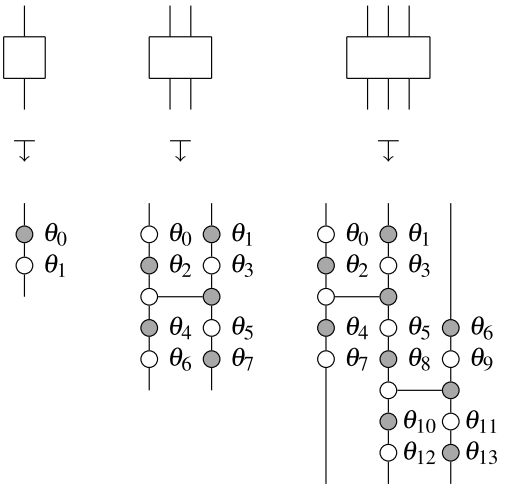


Figure 2.9: CNOT+U(3) ansätze for 1 to 3 qubits. White dots are Pauli-Z-rotations and black dots are Pauli-X-rotations. Cross-connections represent CNOT operations (white dot = control). Graphic taken from [24] under CC BY 3.0 licence.

Hence, we depict the state and effect ansatz for the 3-qubit representation of an ambiguous verb as shown in Figure 2.10.

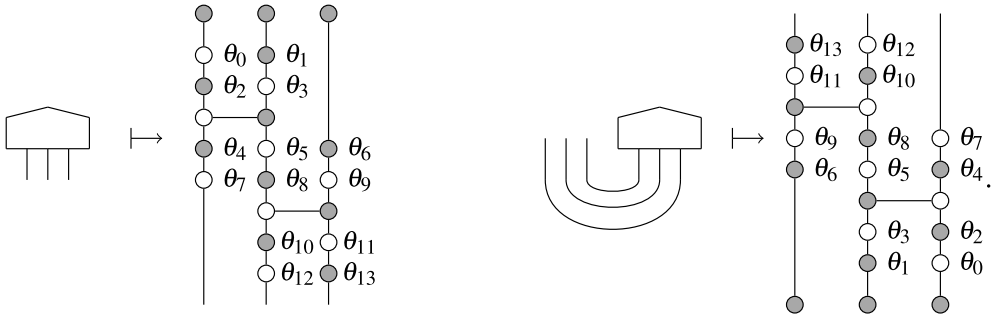


Figure 2.10: CNOT+U(3) Ansatz for ambiguous verbs. Graphic taken from [24] under CC BY 3.0 licence.

Another class of ansätze worth investigating are strongly entangling layers [35]. Originally introduced by Schuld et al. [28], they consist of 1-qubit rotations around all axes for each qubit and CNOT gates between all adjacent qubits. The total amount of parameters to train is then the number of qubits used times 3. An example is depicted in Figure 2.11.

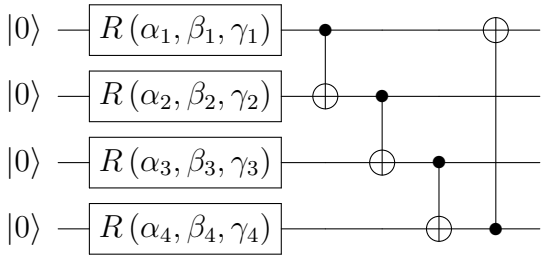


Figure 2.11: Example of strongly entangling layer using four qubits. Using three qubits, we are left with nine parameters per word to optimize, which is significantly less than the 14 used for the 3-qubit CNOT+U(3) ansatz.

Given all the ansätze for the nouns, ambiguous and unambiguous verbs, we then continue to compose the quantum circuits for each phrase and use a suitable optimizer to tune the parameters to reduce the MSE loss. The further procedure of the experiments is described in Chapter 3.

2.3 Modelling abiguity in density matrices

So far, we have shown that it is possible to encode the meaning of ambiguous verbs as superpositions of their single meanings. Although it is evident that this approach has benefits over classical NLP word embeddings (exponential growth of the meaning space by leveraging superposition and entanglement of quantum systems), there are also some drawbacks to mention. To scale the above-introduced toy model to a realistic size, one would need to have a meaning space large enough to encode all possible single meanings of words. Twenty qubits already span a complex Hilbert

space whose dimension surpasses the number of words used in the English language easily. However, it is highly questionable if it is possible to learn any meaningful word representation within such a large vector space. As a side note, one of the currently largest classical NLP models, GPT-3, uses an embedding size of 12.888 [30], while 20 qubits span a complex vector space with dimensionality 1.048.576. To overcome the problem of increasingly large vector spaces Piedeleu et al. [27] and Kartsaklis [19] propose a language model that encodes ambiguous verbs into mixed states described by *density matrices*.

2.3.1 Densitiy matrices

A density matrix ρ describes the statistical state of a quantum system. A quantum system containing various quantum entities can either be of a *pure state* or a *mixed state*. A pure state is characterized by quantum entities that are all in the same state $|\psi\rangle$. Contrarily, a system in a mixed state contains a statistical ensemble of quantum entities in different states $|p_i\rangle$. Each one of these states occurs with a classical probability p_i in the ensemble. The density matrix ρ that characterizes a mixed state is then calculated to be

$$\rho = \sum_{i}^{N} p_i \left|p_i\right\rangle \left\langle p_i\right| \tag{2.11}$$

with $\mathrm{Tr}(\rho) = 1$. Different mixtures of pure states can lead to the same density matrix, hence, they are not distinguishable. Thus, they are physically equivalent. We calculate the expectation value of a measurement according to Born's rule and get

$$\langle A \rangle = \mathrm{tr}(\rho A). \tag{2.12}$$

Using density matrices to model ambiguous verbs has the benefit of using a much smaller vector space to encode word meanings into superpositions and still benefit from other quantum concepts by encoding the meanings of ambiguous verbs into mixed states. Take for example a toy vocabulary of $\{\mathrm{file}, \mathrm{register}, \mathrm{smooth}\}$. Using a quantum language model, we can learn a meaningful vector representation of the abstract senses in a complex Hilbert space, i.e. determine $|\mathrm{register}\rangle$ and $|\mathrm{smooth}\rangle$. Now, instead of learning a non-related vector representation of "file", we encode the meaning into a density matrix of

$$\rho(\mathrm{file}) = p_r \left|\mathrm{register}\right\rangle \left\langle \mathrm{register}\right| + p_s \left|\mathrm{smooth}\right\rangle \left\langle \mathrm{smooth}\right| \tag{2.13}$$

where p_r and p_s are the probabilities of the word "file" to occur in a context where it refers to "register" or correspondingly to "smooth" (within a given text corpus). We then let the density matrix interact with another noun to disambiguate the verb.

2.3.2 Expand the CDM model with density operators

In this section we modify the Compositional Distributional Model of Meaning (CDM) to allow computations using mixed quantum states. So far, we have only used pure

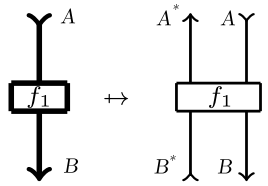
states to represent word meanings. To incorporate density matrices of mixed states into the CDM model, Piedeleu et al. [27] take over the $\mathbf{CPM}(\mathcal{C})$ category by Selinger [29] which is *dagger compact closed* and makes use of morphisms that are **c**ompletely **p**ositive **m**aps (thus, the name CPM). The latter is necessary to ensure that morphisms preserve the positive semi-definite nature of density matrices.

2.3.2.1 From states to operators

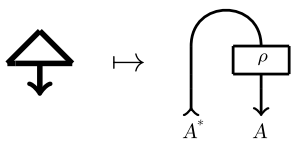
As previously seen in Section 1.3.2, the CDM model relies on a compact closed category where states are seen as morphisms of type $\psi : \mathrm{I} \to A$. However, to allow the system to be described by a density operator, Piedeleu et al. [27] modify the underlying category such that mixed states are described by morphisms $\rho : A \to A$, i.e. linear maps on elements of A. Therefore, they introduce a category $\mathbf{D}(\mathcal{C})$ that inherits all objects from a compact closed category $\mathcal{C}$ and produces completely positive morphisms which are suitable choices for density operators. Speaking in loose terms, the category $\mathbf{D}(\mathcal{C})$ doubles the wires of $\mathcal{C}$:

- Morphisms between objects A and B of $\mathbf{D}(\mathcal{C})$ are morphisms $A \otimes A^* \to B \otimes B^*$ of $\mathcal{C}$.
- An embedding $E : \mathbf{D}(\mathcal{C}) \hookrightarrow \mathcal{C}$, which maps objects and the morphisms from $\mathbf{D}(\mathcal{C})$ to $\mathcal{C}$):
$$E : \begin{cases} A & \mapsto A \otimes A^* \quad \text{on objects;} \\ f & \mapsto f \quad\quad\;\; \text{on morphisms.} \end{cases}$$
- $\mathbf{D}(\mathcal{C})$ is equipped with the same tensor product as $\mathcal{C}$: $\otimes_D = \otimes$.

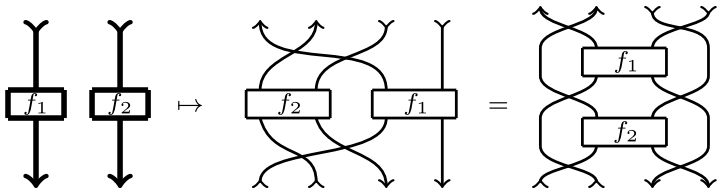
To differentiate between $\mathbf{D}(\mathcal{C})$ and $\mathcal{C}$, they depict diagrams in $\mathbf{D}(\mathcal{C})$ with thick lines. Given the above described axioms about $\mathbf{D}(\mathcal{C})$, they depict a morphism as



In general, word meanings live here in the space of density matrices, which are morphisms that act on objects in $\mathcal{C}$. Hence, we depict the word source in $\mathbf{D}(\mathcal{C})$ as



Furthermore, [27] prove that juxtapose two morphisms in $\mathbf{D}(\mathcal{C})$ (thick lines), corresponds the following operation in $\mathcal{C}$:

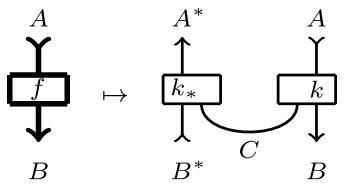


According to Selinger [29], just juxtaposing the morphisms in $\mathcal{C}$ would not necessarily lead to a completely positive morphism in $\mathbf{D}(\mathcal{C})$, which is necessary because only morphisms that are completely positive operatiors take density matrices as arguments and return valid density matrices in the same manner. Finally, Piedeleu et al. [27] define a strictly monoidal functor M that maps the category $\mathcal{C}$ to $\mathbf{D}(\mathcal{C})$ where

$$M : \begin{cases} f_1 \otimes f_2 & \mapsto M(f_1) \otimes_{\mathbf{D}} M(f_2); \\ A & \mapsto A \qquad \text{on objects;} \\ f & \mapsto f \otimes f_* \qquad \text{on morphisms.} \end{cases} \tag{2.14}$$

By defining $f_* = \left(f^\dagger\right)^*$ Selinger [29] proves that $\mathbf{D}(\mathcal{C})$ inherits a †-compact closed structure from $\mathcal{C}$. Piedeleu et al. [27] then continue to adapt the definition of Selinger [29] of a completley positive map, to quote:

> *A morphism $f : A \to B$ of $\mathbf{D}(\mathcal{C})$ is completely positive if there exists an object C and a morphism $k : A \to C \otimes B$, in $\mathcal{C}$, such that f embeds in $\mathcal{C}$ as $(k \otimes k_*) \circ (1_A \otimes \eta_{C^*} \otimes 1_{A^*})$ or, pictorially,*



Denote, that the diagrams from [27] have been adapted to the *top-to-bottom* flow, established in Section 1.3.2[1], which is also consistent with the more detailed dissertation by Kartsaklis [19].

2.3.3 Putting it all together

Now Piedeleu et al. [27] define the category $\mathbf{CPM}(\mathcal{C})$ as a subcategory of $\mathbf{D}(\mathcal{C})$, which inherits the objects and tensor product ($\otimes_{\mathbf{CPM}} = \otimes_{\mathbf{D}}$), but allows only completely positive morphisms. Reducing the functor M, such that the category $\mathbf{CPM}(\mathcal{C})$ only consists of completely positive maps, allows us to write $\tilde{M} : \mathcal{C} \to \mathbf{CPM}(\mathcal{C})$. Furthermore, we define a strong monoidal functor Q that maps the pregroup category $\mathbf{C_F}$ (introduced in Section 1.3.2) to a semantic meaning space $\mathcal{C}$, i.e. giving meaning to a sentence: $Q : \mathbf{C_F} \to \mathcal{C}$. Thus, the compound model of meaning is achieved by the composition:

$$\tilde{M}Q : \mathbf{C_F} \to \mathcal{C} \to \mathbf{CPM}(\mathcal{C}) \tag{2.15}$$

Piedeleu et al. [27] leave it open what kind of category $\mathcal{C}$ could be. Any compact closed category could be to define the semantic space. In terms of quantum computation, the most likely choice is the finite-dimensional Hilbert space category **FHilb**.

[1] Permission to use the graphics has been granted by the authors of [27].

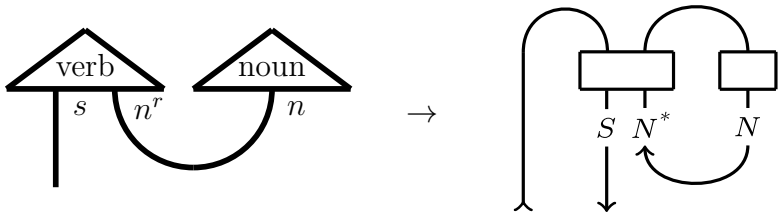
2.3.3.1 From diagrams in CPM($\mathcal{C}$) to diagrams in $\mathbf{C_F}$

Denoting that the atomic types and their duals are mapped from $\mathbf{C_F}$ to $\mathbf{CPM}(\mathcal{C})$ through identity, we are able to use the same diagrammatic notations as described in Section 1.3.4 for sentences that live in $\mathbf{CPM}(\mathcal{C})$. However, the key difference is that single word meanings now live in the space of density matrices instead of a Hilbert space. Hence, Piedeleu et al. [27] constitute the following definition:

> *Let $\rho(w_i)$ be a meaning state $I \to \tilde{M}Q(p_i)$ corresponding to word w_i with type p_i in a sentence $w_1 \dots w_n$. Given a type-reduction $\alpha : p_1 \cdot \dots \cdot p_n \to s$, the meaning of the sentence is defined as:*

$$\rho\left(w_1 \dots w_n\right) := \tilde{M}Q(\alpha)\left(\rho\left(w_1\right) \otimes_{\mathbf{CPM}} \dots \otimes_{\mathbf{CPM}} \rho\left(w_n\right)\right)$$

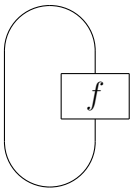
If we now return to our examples from the dataset described in Table 2.1, we can translate the general intransitive verb - noun structure in $\mathbf{CPM}(\mathcal{C})$ to a process diagram in $\mathbf{C_F}$:



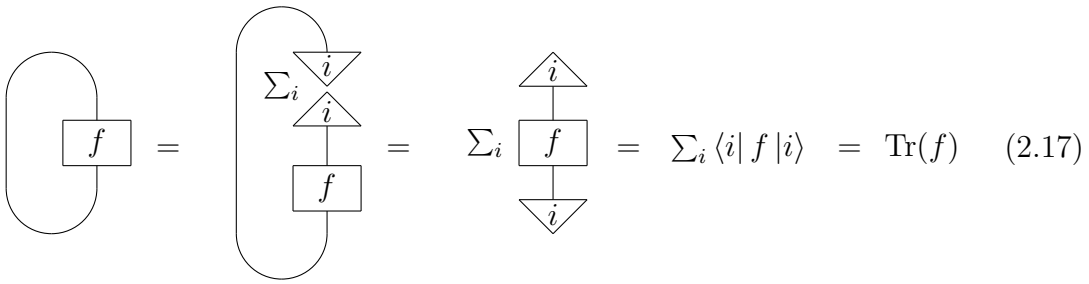
Interestingly, we see that the partial trace of the system describes the composition of two words, which becomes evident after examining the trace in diagrammatic calculus:

2.3.3.2 The trace of a process

Following the diagrammatic reasoning of Coecke and Kissinger [9], we depict the trace of a process as

$$\mathrm{Tr}(f) :=$$  (2.16)

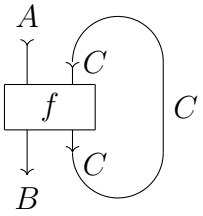
This becomes clear if one inserts the *completeness relation*:



(2.17)

2.3.3.3 The partial trace of a process

Additionally, we can define a partial trace of an operator that lives in a compound Hilbert space $\mathcal{H}_a \otimes \mathcal{H}_b$. Let f be a morphism $f : A \otimes C \to B \otimes C$, then we define the partial trace with respect to the subsystem C as:

$$\mathrm{Tr}_C(f) \coloneqq$$ 

(2.18)

This corresponds to a summation of processes [9], and the output is a matrix whose size depends on the subsystems A and B.

2.3.3.4 Estimating word abiguity

In the CPM model all the words live in the realm of density matrices. Therefore, Kartsaklis [19] suggests to measure the knowledge about an open quantum system by calculating the *von Neumann entropy* of the density matrix:

$$S(\rho) = -\mathrm{Tr}(\rho \ln \rho) \tag{2.19}$$

The value of $S(\rho)$ drops to zero if ρ represents a pure state and increases for mixed states up to a value of $\ln(D)$, which is the dimension of the underlying Hilbert space. Therefore, the von Neumann entropy is a suitable measurement to evaluate the ambiguity of a word and how context disambiguates it. We, therefore, continue to present a workflow to evaluate the proposed CPM model for the given disambiguation task.

2.3.4 Proposed disambiguation workflow

We will further work with a hybrid model. The benefit of such a hybrid model is that, for low-dimensional Hilbert spaces, we can analyze quantum states and density matrices classically and demonstrate the general possibilities of the CPM model. The dataset introduced by Kartsaklis, Sadrzadeh, and Pulman [18] contains 4 ambiguous verbs of two distinct meanings each, and 35 context words, i.e. nouns (see Table 2.1). We are given the vector representation of the context words as four-dimensional vectors. These vectors are mined from the English Wikipedia through the project Wikipedia2Vec [36] and underwent a dimensionality reduction through PCA (for the full list of vectors, see Appendix A.2). We then use the Compositional Distributional Model of Meaning (CDM) to learn the vector representation of the unambiguous verbs (*register*, *smooth*, ...) such that the dataset is well-approximated. Due to the four-dimensional vector representation, it is sufficient to use only two qubits for our model (the Hilbert space of two qubits has $2^2 = 4$ dimensions). This procedure follows closely the methods described in Section 2.2.1.
So far, the usage of quantum computers is justified by the exponentially growing vector space in which the words are embedded. However, after fitting the model, we

can construct the density matrices for each ambiguous verb (*file*, *dribble*, ...) and investigate their interaction with the context words according to the CPM model to explore the potential of word meanings based on mixed states.

3
Methods

This chapter will present the methods used to translate the proposed language models into quantum circuits and describe the procedure to fit the models. The justifications for the used methods are laid out accordingly.

3.1 Disambiguation based on Superpositions

First, we will cover the implementation of the ambiguity model presented in Section 2.2. For this task, we exploit the rich features of the `DisCoPy` python package [15] which is tailored to simplify the handling of monoidal categories and interacts smoothly with the t|*ket*⟩ quantum circuit compiler [31].

3.1.1 1-qubit Model

As described in Section 2.2.1, we map each word type to a variational ansatz. Therefore, we build a functor that maps the monoidal category diagram into a circuit as in Figure 2.4, which yields a *QCirc* python object. However, to compute the result of the circuit, we need to compile the sentence-specific instance of QCirc into code that a Quantum device can process. This is easily done by using the t|*ket*⟩ compiler, which returns the circuit shown in Figure 3.1

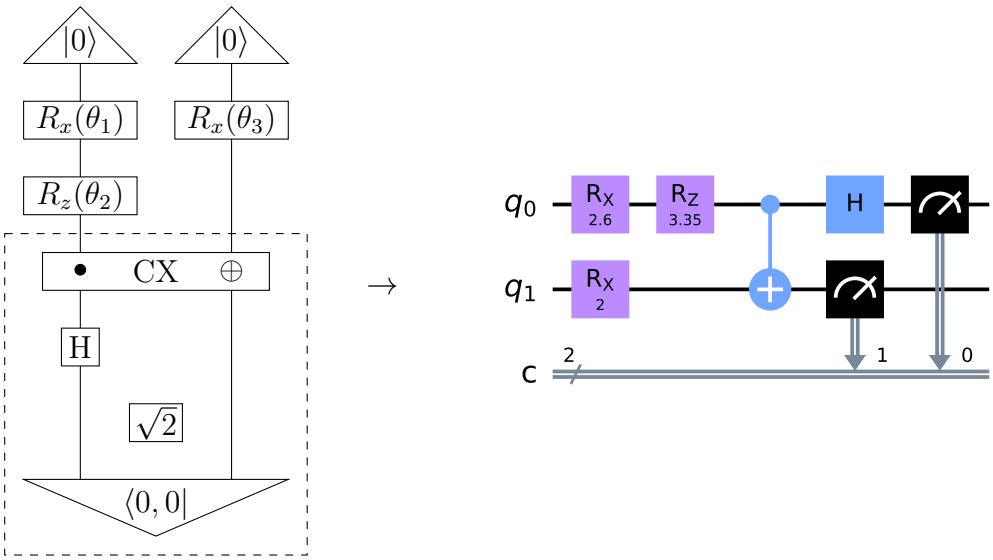


Figure 3.1: Using t|*ket*⟩ to compile a QCirc instance to a Qiskit [1] model.

It is essential to mention that the $\langle 0,0|$ effect results in a post-selection measurement. The number of fulfilled post-selections divided by the number of circuit-shots then equals the "truthness" of the sentence. The cup ansatz (highlighted in Figure 3.1 by the dashed box) checks for parity of the two qubits. However, in the case of parity, the probability that the code returns $|0,0\rangle$ is only $1/2$. Hence, we need to scale the amplitude by $\sqrt{2}$, which is indicated by the $\boxed{\sqrt{2}}$ box in the QCirc diagram. t$|ket\rangle$ interprets this as scaling the post-selection result by 2.

3.1.1.1 Optimization

Parsing all sentences from the dataset associated with the ambiguous verb "file" into diagrams and eventually into quantum circuits leaves us with nine parameters for the nouns and two parameters for the ambiguous verb itself. Running the model for all 27 sentences and computing the MSE gives a metric for optimization. In a classical setting, one could use gradient-based methods to reduce the MSE. However, there is no known way to perform backpropagation on a quantum device (so far). Hence, computing gradients by the parameter shift rule is expensive. We, therefore, decide to use a simple genetic algorithm to find a first estimate of the optimal parameter set. Genetic algorithms use concepts from biological evolution, like natural selection, mutation, and crossover, to find the solution to a problem. Detailed information about the algorithm and the actual python implementation can be found in [32].

Finetuning After using the genetic algorithm to find a good approximation of the parameters, we use a noisy optimization method to finetune the result. In general, a noisy optimization method seeks to solve a problem that is formulated as

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) = \min_{\boldsymbol{x}} \mathbb{E}_{\xi}[F(\boldsymbol{x}, \xi)]. \tag{3.1}$$

Running an algorithm $f(\boldsymbol{x})$ on a (Noisy intermediate-scale) quantum computer currently does not allow us to access the result directly. Due to a high level of noise, the only option is to compute the expectation value of the noisy function $F(\boldsymbol{x}, \xi)$, where ξ indicates the noise term. To solve this problem, we use the *Simultaneous Perturbation Stochastic Approximation* (SPSA) algorithm developed by Spall [33]. The SPSA algorithm estimates the gradients through a probabilistic gradient shift method and updates the parameters through a decreasing learning rate. It is designed to optimize noisy functions, where we only have access to the expectation value of the underlying function. The details and pseudocode of the SPSA implementation are found in [33]. The validity of this method for quantum circuits has been proven by Meichanetzidis et al. [24], as they could fit a small QNLP model on an actual NISQ device, using SPSA.

3.1.1.2 Quantum Simulation Backend

One benefit of the t$|ket\rangle$ is that code produced by the compiler is hardware-agnostic, i.e., runs on many different devices, as well as through the quantum simulators provided by the Qiskit software [1]. Theoretically, a validation of the introduced models through actual quantum computers can be achieved. However, we limit

ourselves to simulations to avoid complications due to erroneous computations of NISQ devices.

3.1.2 3-qubit Model

After validating the approach by running the 1-qubit model to encode the ambiguous verb "*file*", we expand the model to encode all 4 ambiguous verbs of the dataset at once ("*file*", "*dribble*", "*tap*", "*charge*"). As previously described, these words entail eight abstract senses. Hence, we use the computational basis states of 3 qubits to encode the single meanings as described in Section 2.2.3. Fitting the model works the same as for the 1-qubit model. However, due to the larger dataset, we skip the pre-fitting with the genetic algorithm and feed the SPSA algorithm with batches of 4 sentences. Using the CNOT+U(3) ansatz for the ambiguous verbs (see Figure 2.9) and the noun ansatz depicted in Figure 2.8a leaves us with $4 \times 14 + 35 \times 3 = 161$ parameters to optimize.
An example circuit is shown in Figure 3.2.

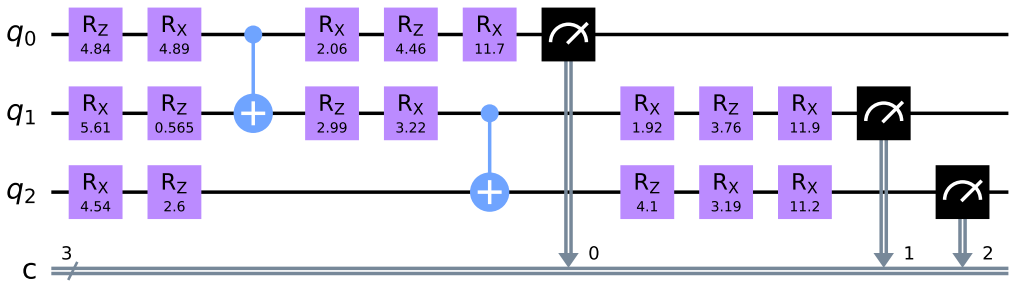


Figure 3.2: Qiskit circuit for "*file nail*" using 3 qubits and the CNOT+U(3) ansatz. Post-selecting on the $\langle 0, 0, 0|$ effect yields the result of the circuit calculation. A higher percentage of fulfilling the post-selection condition equals a higher plausability score for the sentence.

3.1.3 Evaluation

To evaluate the proposed models, we adapt the metrics from Grefenstette and Sadrzadeh [14] and Piedeleu et al. [27]. They divide the dataset into sentences of low plausibility (e.g. "*register nail*", score 0) and high plausibility (e.g. "*smooth nail*", score 1) and calculate the cosine similarity of the model's outcome and the expected result. While Grefenstette and Sadrzadeh [14] compute a 2-dimensional similarity array, our proposed quantum model calculates only a 1-dimensional score. Therefore, in our case, the cosine similarity equals the calculated score from the model after recentering the ground truth to one. Furthermore, they calculate the correlation between the ground truth and the predictions using Spearman's rank correlation coefficient ρ. Spearman's ρ is a nonparametric measure that indicates that a monotonous function exists that maps a variable to another. The benefit of using Spearman's ρ is that it does not assume any underlying distribution of the

data, which makes it versatile but also ruthless. The test result is within a range of $[-1, 1]$, where a value around zero indicates a low correlation.

3.2 Disambiguation based on density matrices

In order to implement the disambiguation method described in Section 2.3.2, we first have to propose a quantum representation of the abstract senses in the dataset, i.e., the senses of the ambiguous verbs and the nouns. To resemble a real-world application, we first determine vector representations of the nouns from a large text corpus. These vectors (if not too large) can be initialized on a quantum circuit using a QRAM method. Note that, in general, QRAM becomes unfeasible for large vectors. Afterward, we fit the verb states using QNLP methods, i.e., we define variational quantum ansätze, implement a loss function, and optimize the circuits using the SPSA optimizer. We close this chapter by briefly discussing the quantum circuit for the compositional CPM model of our verb-noun phrases and evaluate the proposed model using classical counter-parts of the quantum model.

3.2.1 Determining the noun representations

To resemble a real-world scenario, we seek to determine vector representations for each context word (i.e., nouns) from a large text corpus. The most common word vector techniques come from the *word2vec* algorithm family [25]. Most word2vec algorithms use either continuous bag-of-words (CBOW) or skip-grams combined with neural networks to embed word meanings into a high-dimensional vector space. By defining a fixed sliding window that passes the whole text corpus, the neural network takes a target word and predicts the context words (skip-gram) or takes the context words and predicts the target word (CBOW). The neural network used for this task is commonly an encoder-decoder network, and after successfully training the algorithm, we can cut off the decoder. The encoder then produces vectors (embeddings) for each unique word. Training such a model requires a lot of computational resources. Hence we help ourselves by using pre-trained word embeddings based on the English Wikipedia from 2018 provided by Yamada et al. [36].

3.2.1.1 Dimensionality reduction

The lowest-dimensional word embeddings from [36] are of size $d = 100$. We use principal component analysis (PCA) to reduce the dimension to $d = 4$. PCA computes the eigenvectors of the covariance matrix of the data and spans a new orthonormal basis using those that correspond to the largest eigenvectors. Discarding all eigenvectors with small eigenvalues lets us reduce the dimensionality while ideally retaining the main characteristics of the data distribution. The reduced vectors for all nouns in the dataset are found in the Appendix A.2. Some examples are

account	[-0.915559	-0.290692	-0.023149	0.276973]
application	[-0.628319	-0.341554	-0.677950	-0.170117]
ball	[0.744622	0.504402	0.344781	-0.268779]

3.2.1.2 Initialize the noun states using QRAM

After mining the noun vectors from a large text corpus and applying dimensionality reduction, we seek to initialize the nouns as quantum states. Theoretically, we can encode an n-dimensional vector in $\log_2(n)$ qubits, which means that in our case two qubits suffice. Given a normalized data array $\mathbf{a}$, we seek a quantum circuits that initializes the following state:

$$|\mathbf{a}\rangle = \frac{1}{\|\mathbf{a}\|} \sum_{i=1}^{n} a_i |i\rangle . \tag{3.2}$$

A method to initialize such a state is presented by Kerenidis and Prakash [20]. Having a 4-dimensional data array $\mathbf{a} = [a_1, a_2, a_3, a_4]$, with $\|\mathbf{a}\|_2 = 1$, we first apply a rotation on the first qubit:

$$|0\rangle |0\rangle \rightarrow \left(\sqrt{\|[a_1, a_2]\|} |0\rangle + \sqrt{\|[a_3, a_4]\|} |1\rangle \right) |0\rangle \tag{3.3}$$

Consequently, we apply another rotation conditioned on the first qubit. Hence, we generate

$$\begin{aligned} &\left(\sqrt{\|[a_1, a_2]\|} |0\rangle + \sqrt{\|[a_3, a_4]\|} |1\rangle \right) |0\rangle \rightarrow \\ &\sqrt{\|[a_1, a_2]\|} |0\rangle \frac{1}{\sqrt{\|[a_1, a_2]\|}} (a_1 |0\rangle + a_2 |1\rangle) + \\ &\sqrt{\|[a_3, a_4]\|} |1\rangle \frac{1}{\sqrt{\|[a_3, a_4]\|}} |1\rangle (a_3 |0\rangle + a_4 |1\rangle) \\ &= a_1 |00\rangle + a_2 |01\rangle + a_3 |10\rangle + a_4 |11\rangle . \end{aligned} \tag{3.4}$$

The rotation we use is the parametrized $R_y(\theta)$ gate:

$$R_y(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} . \tag{3.5}$$

The $R_y(\theta)$ gate takes the $|0\rangle$ state to the following superposition:

$$R_y(\theta) : |0\rangle \rightarrow \cos\left(\frac{\theta}{2}\right) |0\rangle + \sin\left(\frac{\theta}{2}\right) |1\rangle . \tag{3.6}$$

To find the right angle θ for the rotations in Equation 3.4 and Equation 3.5 we can use the inverse of the cosine using the data array $\mathbf{a}$.

3.2.2 Verb ansatz and Training

As we intend to learn to represent the phrases from the dataset within the `DisCoPy` framework, we need to use a suitable parametrized ansatz for the verbs. Therefore, a suitable choice is a strongly entangling layer presented in Figure 2.11 for two qubits. Following Section 3.1, we translate the compositional model into a quantum circuit, evaluate it for each phrase in the dataset and compute the mean squared error as

a loss function. To reduce the loss, we use the SPSA algorithm, which we covered in Section 3.1.1.1. In contrast to the previously described models, we only learn the quantum states of the verbs that define the *abstract senses* of the ambiguous verbs (i.e., "*register*", "*smooth*", "*carry*", ...). That allows us later to construct the ambiguous verbs as mixed states of the quantum states of their senses.

3.2.3 Comparison to the baseline

Since we previously generated the noun vectors from a text corpus using classical NLP methods, it is viable to determine the vector representation of the verbs classically too. This comes in handy as we can use the classical result as a benchmark for the quantum model. In an ideal world, the nouns would also be mined using QNLP methods, including many qubits, which eventually makes a classical evaluation unfeasible.
Like quantum mechanics in general, the Compositional Distributional Model of Meaning is a purely linear model. Hence, we can combine the noun vectors into a matrix N and solve a linear system to determine the verb matrix V according to

$$VN = S, \tag{3.7}$$

where S bundles the sentence vectors of the noun-verb-pairs. The noun matrix N combines all noun-vectors as column vectors $[\overrightarrow{n}_1, \overrightarrow{n}_2, \dots]$, while the verb matrix V consists of the verbs as row-vectors $[\overrightarrow{v}_1^{\mathrm{T}}, \overrightarrow{v}_2^{\mathrm{T}}, \dots]$. Reducing the dataset to the ambiguous verb "*file*" and the two nouns "*account*" and "*smooth*" yields the following minimal example:

$$VN = \begin{bmatrix} \overrightarrow{\text{register}}^{\mathrm{T}} \\ \overrightarrow{\text{smooth}}^{\mathrm{T}} \end{bmatrix} [\overrightarrow{\text{account}}, \overrightarrow{\text{nail}}] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = S. \tag{3.8}$$

This system of equations can generally not be solved exactly, which is why we calculate the least-squares solution by using the Moore–Penrose inverse N^+ of the noun matrix:

$$V = SN^+. \tag{3.9}$$

This procedure can be repeated for each of the ambiguous verbs and all of their associated nouns. Hence, we solve four least-squares systems to retrieve the vectors of the eight sense verbs. Note that we have to normalize the calculated vectors to make them comparable to the quantum states retrieved by the QNLP model. The final MSE provides a benchmark for evaluation.

3.2.4 Sentence composition within the CPM model

Investigating the composition words initialized as a mixed state on a circuit model is out-of-scope for this thesis. Possible methods are given by Coecke and Meichanetzidis [10], namely the "*fuzz*" and the "*phaser*" operation. A model to initialize mixed states on a circuit model is given in the Appendi A.4. To simplify the analysis, we continue with a hybrid approach, where we extract the state vectors of the sense

verbs from the `DisCoPy` quantum simulation and calculate the density matrices of the ambiguous verbs classically. The following workflow is based on the work of Kartsaklis [19] and is quantum-inspired, i.e., provides an intuition for the power of the CPM model.

3.2.5 Classical composition of words as density matrices

As previously seen, we can represent (ambiguous) words as mixtures of their sense vectors, which are described by density matrices:

$$\rho(\text{word}) = \sum_{i=1}^{n} p_i \left|\text{sense}_i\right\rangle \left\langle\text{sense}_i\right| . \tag{3.10}$$

Kartsaklis [19] then proposes a model of composition by applying the Hadamard product (point-wise multiplication) between two words represented as density matrices. Hence,

$$\widetilde{\rho}_{\text{comp}} = \rho_{\text{verb}} \odot \rho_{\text{noun}}. \tag{3.11}$$

Kartsaklis [19] derives this model from the axioms of the Compositional Distributional model of meaning, including terms of a Frobenius algebra. For further details, we reference Chapter 7.2 of [19]. Importantly, we have to remark that the Hadamard product of two density matrices is not trace-preserving. Hence, we have to normalize the resulting matrix, such that it has trace one:

$$\rho = \frac{\widetilde{\rho}}{\text{Tr}(\widetilde{\rho})}. \tag{3.12}$$

To estimate the ambiguity of the verb-noun phrase, we calculate the von Neumann entropy S of the compound density matrix (see Section 2.3.3.4). Ideally, composing an ambiguous verb with a context word (noun) would lower the entropy to zero, which represents an unambiguous statement. Therefore, we construct the density matrix of the ambiguous verbs as

$$\rho_{\text{amb}} = \frac{1}{2} \left|\text{sense}_1\right\rangle \left\langle\text{sense}_1\right| + \frac{1}{2} \left|\text{sense}_2\right\rangle \left\langle\text{sense}_2\right| , \tag{3.13}$$

while the density matrix of the context word is simply

$$\rho_{\text{noun}} = \left|\text{noun}\right\rangle \left\langle\text{noun}\right| . \tag{3.14}$$

We can then record the *disambiguation power* P of a noun by

$$P = \frac{S(\rho_{\text{amb}}) - S(\rho_{\text{comp}})}{S(\rho_{\text{amb}})}. \tag{3.15}$$

To validate the model, we also need to check the disambiguation power of random context words, i.e., randomly initialized noun vectors. We can then place the outcome of the proposed model next to the randomized baseline group and determine if the proposed model disambiguates the verbs with statistical significance. A suitable method to compare the difference in distribution is Wilcoxon's rank-sum test.

3.2.5.1 Similarity measure of disambiguated verbs

Besides evaluating the disambiguation power of nouns, we can compute the similarity between the disambiguated verb and the according sense verb. According to Piedeleu et al. [27], the similarity between two words represented as density matrices ρ_1 and ρ_2 can be computed by the following:

 $\rightarrow \quad \mathrm{Tr}(\rho_2^\dagger \rho_1).$ (3.16)

Hence, we can compute the similarity between word senses and disambiguated verbs and test for statistical significance. Take the example "*dribble ball*": Constructing the density matrix for *dribble* yields

$$\rho_{\text{dribble}} = \frac{1}{2} |\text{drip}\rangle \langle \text{drip}| + \frac{1}{2} |\text{carry}\rangle \langle \text{carry}| , \tag{3.17}$$

while the unambiguous context word is simply

$$\rho_{\text{ball}} = |\text{ball}\rangle \langle \text{ball}| . \tag{3.18}$$

Further, we calculate the composition of ρ_{dribble} and ρ_{ball} according to Equation 3.11 and 3.12, denoted as ρ_{comp}. To test the success of the disambiguation, we now calculate the similarity between ρ_{comp} and the density matrices of the possible verb senses ρ_{carry} and ρ_{drip}. In this example, ideally, the similarity of ρ_{comp} to ρ_{carry} is larger than the similarity to ρ_{drip}. Evidently, we need to compare the similarities of proper disambiguated verbs to its target sense compard to a randomized baseline and perform Wilcoxon's rank-sum test for each ambiguous verb to prove a statistically significant disambiguation. We refere to Section 4.2.2 for more details.

4

Results

In this chapter, we present the results of the experiments described in Chapter 3. We start by describing the outcome of the minimal 1-qubit model (Section 3.1.1), where we have reduced the dataset only to contain one ambiguous verb. We then continue with the analysis of the experiment using the whole dataset, i.e., the 3-qubit model (Section 3.1.2). Last but not least, we present the results of the CPM model of disambiguation based on density matrices.

4.1 Superposition model

In this section, we describe the results of the superposition model from Section 3.1.

4.1.1 1-qubit Model

The following paragraphs describe the model fit and the statistical results of the 1-qubit minimal example presented in Section 3.1.1.

Fitting the model To estimate a first good fit of the model's 19 parameters, we first use a genetic algorithm (GA) with a population size of 5, 20 generations, and a mutation probability of 0.1. In Figure 4.1 we record the mean-squared error (MSE) over the GA generations.

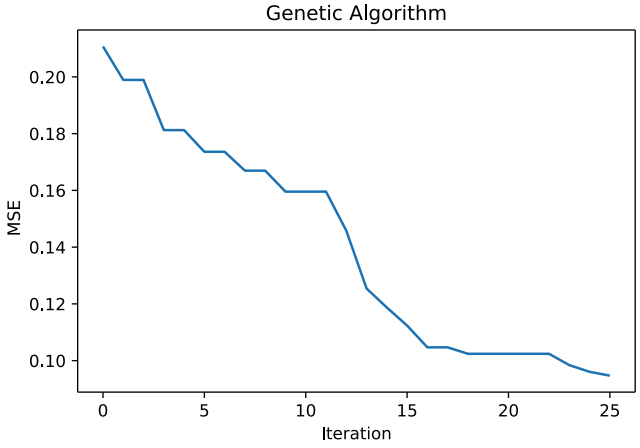


Figure 4.1: GA fit of the 1-qubit model. We observe a rapid decline of the loss function during the first few iterations.

We observe that after a few initial steps, the MSE decreases to a value of 0.076, which is a suitable initial value for the SPSA algorithm. Investigations using a genetic algorithm with more generations and larger population sizes have shown that the fitting process becomes increasingly unpractical after a few iterations. The probability of reducing the loss through random mutations and cross-overs shrinks over time. Hence, it is preferable to continue using a gradient-based method for finetuning, like the SPSA algorithm. The result of the fit is shown in Figure 4.2.

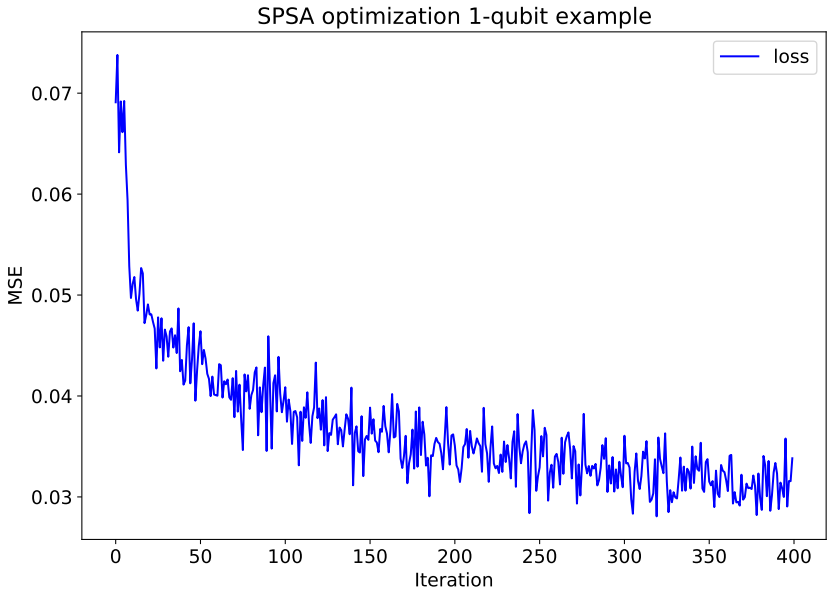
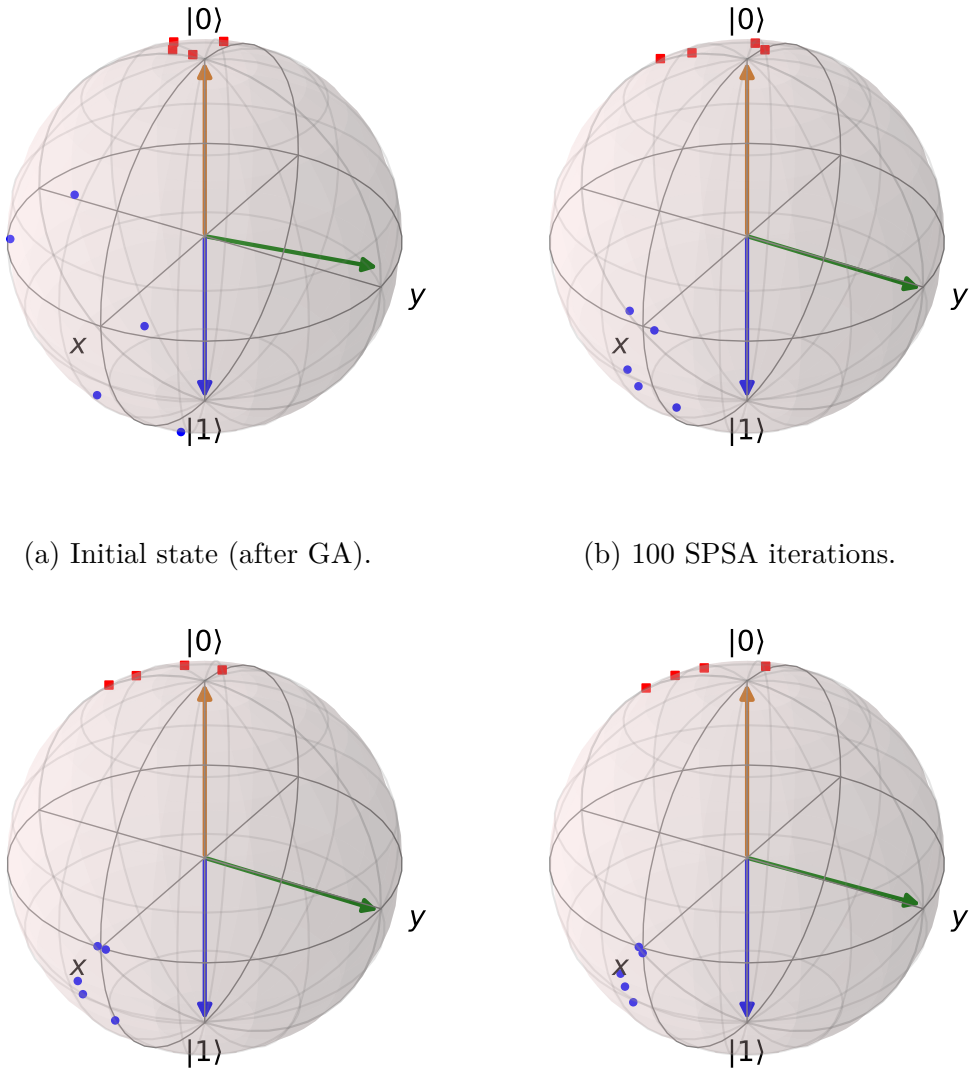


Figure 4.2: SPSA fit of the 1-qubit model. We observe the noisy nature of the loss function. The loss appears to level out at a value of around 0.03.

The SPSA algorithm has two hyper-parameters: The initial learning rate A and the initial gradient-shift step size B. Without thoroughly investigating the optimal parameters, the values $A = 0.2$ and $B = 0.1$ achieved good results. The final loss is 0.0307.

4.1.1.1 Results

A benefit of having a 1-qubit minimal example is that we can visualize the state vectors on a Bloch sphere (see Figure 4.3). The two verbs *smooth* and *register*, which mark the senses of the ambiguous word *file*, denote the poles of the Bloch sphere ($|\text{smooth}\rangle = |0\rangle$ and $|\text{register}\rangle = |1\rangle$). The noun states are visualized as dots on the bloch sphere. As expected, we see that the nouns align according to their usage in the text corpus; nouns that occure with *smooth* (*nail*, *steel*, ...) are drawn towards the $|0\rangle$ pole, and those who occure with *register* (*account*, *application*, ...) align close to the $|1\rangle$ pole. That confirms the underlying distributional assumption of language, i.e., words with similar meaning remain close within the vector space.



(a) Initial state (after GA).

(b) 100 SPSA iterations.

(c) 200 SPSA iterations.

(d) 400 SPSA iterations.

Figure 4.3: Evolution of the word states during the SPSA fit. Verb states are depicted as vectors and noun states as points. Red dots indicate nouns that occur in the context of *smooth* and blue dots indicate nouns that occur in the context of *register*. The green arrow indicates the quantum state of the word *file*.

To further investigate the results, we split the dataset in three parts: Sentences with low co-occurrence score (*register nail*, 0), sentences with high co-occurrence score (*smooth nail*, 1), and sentences containing the ambiguous verb (*file nail*, 1). We then compare the similarity of the annotated score (binary) to the prediction of the model (see Figure 4.4). We see that the model performs fairly well for sentences containing abstract senses register and *smooth*. Visually this is explained by the position of the noun states on the Bloch sphere (see Figure 4.3 (d)). All nouns are in close vicinity to the poles; thus, they compose well with the states of the abstract senses $|0\rangle$ and $|1\rangle$.

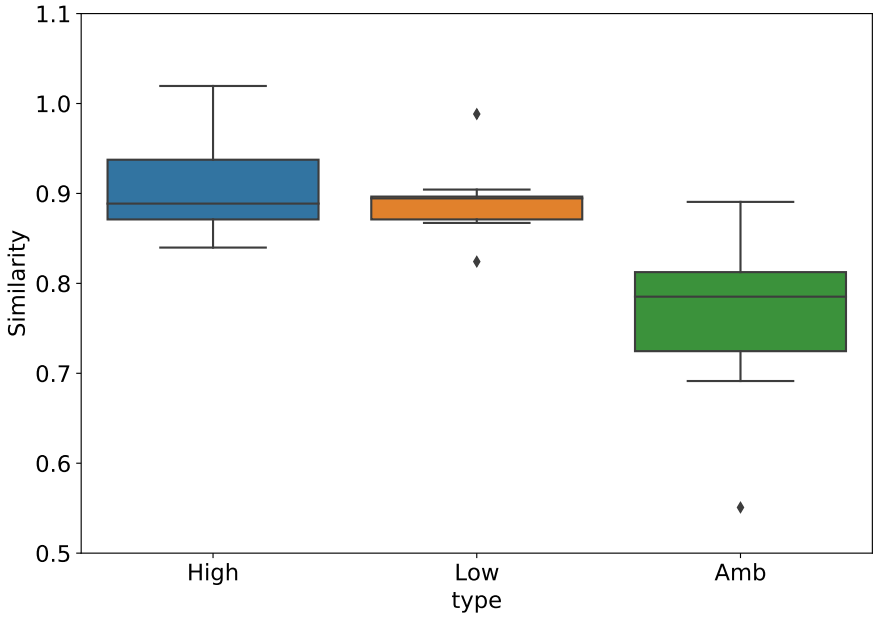


Figure 4.4: Similarity scores of the ground truth and the predicted score for the 1-qubit model. The dataset is split into three parts (low/ high co-occurrence of words and sentences containing the ambiguous verb). A score of 1 equals a perfect prediction. Scores above 1 are due to probabilistic artifacts of the quantum simulator.

However, looking at the predicted scores of the model on sentences containing the ambiguous verb, we see a clear drop in performance. We find that the quantum state of *file* is

$$|\text{file}\rangle = -0.714\,|\text{smooth}\rangle - 0.700i\,|\text{register}\rangle \tag{4.1}$$

which confirms the initial assumption that $|\text{file}\rangle$ is a superposition of the abstract senses, where the modulus squared of the amplitudes is approximately 0.5. However, that also means that we can not achieve a perfect score with this model, as a noun state that composes perfectly with $|\text{register}\rangle$ would only return a score of 0.5 if composed with $|\text{file}\rangle$. A solution to this problem would be to scale the quantum state of $|\text{file}\rangle$ such that the modulus squared of the amplitudes is approximately 1, or to introduce a non-linear activation function applied to the outcome of the quantum model. For the following 3-qubit model on the full dataset we decide to scale the quantum states of the ambiguous verbs accordingly.

Finally, we record a Spearman's rank correlation coefficient ρ between the predictions and the human annotations of $\rho = 0.8171$ ($p < 0.01$), indicating a high correlation between the model's predictions and the true values.

4.1.2 3-qubit Model

Here we examine the results of the 3-qubit model used on the whole dataset. The theory is laid out in Section 3.1.2. Note that the process differs slightly from the 1-qubit model. Due to the size of the dataset, we feed the SPSA algorithm with batches of size 4, where we separate the fitting process of the nouns and the ambiguous verbs. Furthermore, we investigate two different ansätze for the ambiguous verbs (see Figure 2.10 and 2.11). Note that we scale the amplitude of the ambiguous verbs by a factor of $\sqrt{2}$ to guarantee an optimal fit of the model.

Fitting the model To facilitate the fitting process, we first fit the parameters of the noun ansätze. The dataset contains 35 nouns, and three parametrized rotations define each noun. Hence, there are 105 parameters to optimize. Fitting a larger amount of parameters is optimally done by feeding the SPSA algorithm with batches. The MSE loss over the epochs is presented in Figure 4.5.

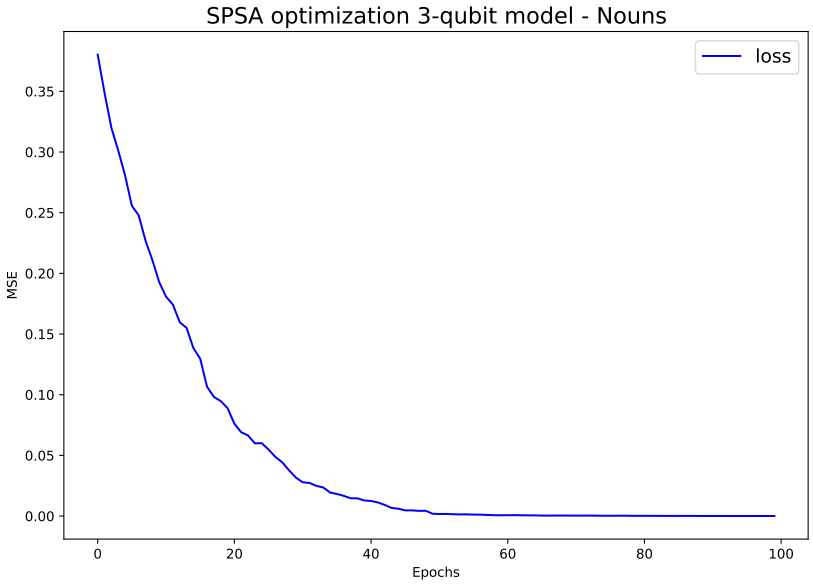


Figure 4.5: SPSA fit of the 3-qubit model - nouns only. The process is seemingly less affected by noise, which is due to the recording of the data after each epoch. Each epoch consits of many SPSA iterations, which compensates for the original noise.

As we can see, the nouns can be represented perfectly within the model. This is due to the simple composition of the model and the fact that each noun only shares high similarity with one verb sense. Consider the example of the sense verb *accuse*: In our model, *accuse* is represented by the basis state $|100\rangle$. Hence, all nouns that compose with *accuse* (e.g. *offender*, *defendant*) are optimally represented by the $\langle 100|$ effect, which is just a π-rotation (bit-flip) on the first qubit, and is a rather simple scenario to learn.

As previously mentioned, we seek to compare the performance of the two ansätze used for the ambiguous verbs, which is why we have separated the learning process of the nouns and the verbs. The MSE loss curve of the verb process is shown in Figure 4.6.

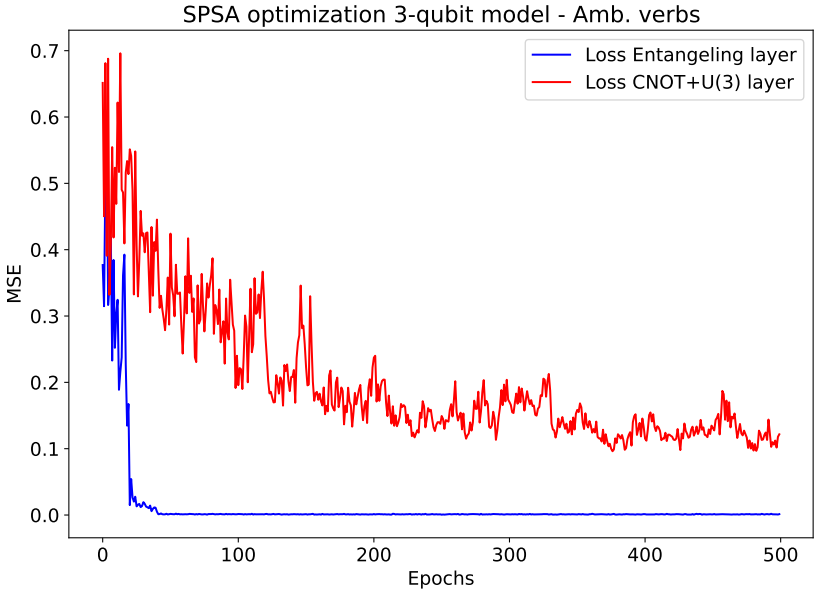


Figure 4.6: SPSA fit of the 3-qubit model - ambiguous verbs only.

We can see that using the Strongly Entangling Layer as verb ansatz yields a far better fit than using the CNOT+U(3) ansatz. Within a few SPSA steps, we see that the loss for the Strongly Entangling Layer experiment drops to zero, while it appears that the CNOT+U(3) layer cannot generalize correctly. When using the CNOT+U(3) layer, the loss levels out at a value around 0.1, which is not satisfying, given that the model should fit the data perfectly. One possible reason for the performance discrepancy of both ansätze is that the Strongly Entangling Layer only uses nine parameters instead of the 14, which the CNOT+U(3) ansatz uses. Additionally, it may be impossible to reach all superpositions using the CNOT+U3 ansatz. However, the last assertion is not validated and would require a thorough analysis.

4.1.2.1 Results

Following the analysis of the 1-qubit model, we record the similarities values of the model's predictions compared to the ground truth. Recall that a value of 1 is the optimal score, and values that are slightly above one occur due to statistical artifacts from the quantum model in combination with the amplitude scaling factor of $\sqrt{2}$. The results for the 3-qubits model using the Strongly Entangling Layer are displayed in Figure 4.7

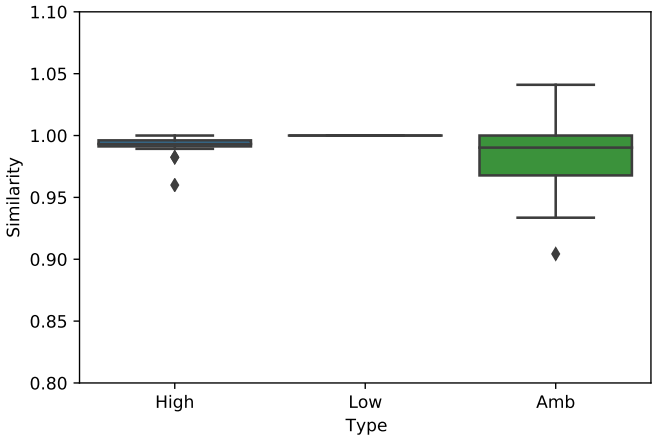


Figure 4.7: Similarity scores for the 3-qubit model using the Strongly Entangling Layer as ambiguous verb ansatz. The results are separated into sentences with high word co-occurrence ("*register account*"), low word co-occurrence ("*register nail*"), and sentences containing ambiguous verbs ("*file nail*").

As already indicated by the fitting curves in Figure 4.5 and Figure 4.6, the model captures the data excluding the ambiguous verbs quite perfectly. Interestingly, we observe a small error bar for sentences with high degree of truth, while sentences with low degree of thrut are captured perfectly. That may be caused by sentences with high word co-occurrences, where the noun and verb ansätze have to match perfectly (for example $|\text{verb}\rangle = |\text{noun}\rangle = |000\rangle$), while for sentences with low word co-occurrences any non-equal combination of basis states would return a measurement result of zero.

Comparison of CNOT+U(3) and Strongly Entangling ansätze Plotting the results for sentences containing the ambiguous verbs for both different ansätze confirms that the Strongly Entangling ansatz is more versatile and captures the data better.

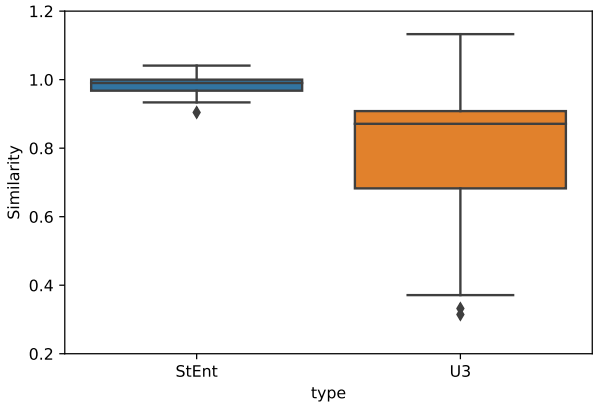


Figure 4.8: Comparison of CNOT+U(3) and Strongly Entangling ansätze for ambiguous verbs.

Using the strongly entangling ansatz, we achieve a median similarity score of 0.99, while the CNOT+U(3) ansatz only yields a median score of 0.87. Considering the fitting process in Figure 4.6, we conclude that the data indicate substantial advantages of the strongly entangling ansatz over the CNOT+U(3) ansatz. Thus, for all further experiments, we use only the strongly entangling ansatz.

4.1.2.2 Comparing the verb state amplitudes

We have encoded the unambiguous verb-sense into the basis states of the 3-qubit system. Hence, we analyse the state vectors of the fitted ambiguous verbs and verify that the superpositions contain the correct verb sense (see Equation 2.6). Each ambiguous verb is described by a 8-dimensional complex vector that lives in a Hilbert space:

$$|\text{amb-verb}\rangle = \begin{pmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{pmatrix} \begin{matrix} \text{knock} \\ \text{carry} \\ \text{drip} \\ \text{smooth} \\ \text{accuse} \\ \text{intercept} \\ \text{register} \\ \text{bill} \end{matrix} \tag{4.2}$$

After fitting the model, we record

$$|\text{file}\rangle = \begin{pmatrix} 0. - 0.02i \\ 0.01 + 0.i \\ -0. + 0.i \\ -0.64 + 0.31i \\ -0.01 + 0.01i \\ -0.01 + 0.01i \\ 0.65 + 0.28i \\ 0. - 0.i \end{pmatrix} \begin{matrix} \_ \text{ knock} \\ \_ \text{ carry} \\ \_ \text{ drip} \\ \blacksquare \text{ smooth} \\ \_ \text{ accuse} \\ \_ \text{ intercept} \\ \blacksquare \text{ register} \\ \_ \text{ bill} \end{matrix} \qquad |\text{charge}\rangle = \begin{pmatrix} 0. + 0.01i \\ 0. + 0.i \\ 0. - 0.i \\ -0.01 + 0.i \\ -0.15 + 0.69i \\ 0. + 0.i \\ -0. + 0.i \\ 0.66 - 0.26i \end{pmatrix} \begin{matrix} \_ \text{ knock} \\ \_ \text{ carry} \\ \_ \text{ drip} \\ \_ \text{ smooth} \\ \blacksquare \text{ accuse} \\ \_ \text{ intercept} \\ \_ \text{ register} \\ \blacksquare \text{ bill} \end{matrix}$$

$$|\text{dribble}\rangle = \begin{pmatrix} 0. + 0.i \\ 0.22 - 0.67i \\ 0.11 - 0.7i \\ 0. + 0.i \\ -0. + 0.i \\ -0.02 + 0.01i \\ -0.02 + 0.i \\ -0. + 0.i \end{pmatrix} \begin{matrix} \_ \text{ knock} \\ \blacksquare \text{ carry} \\ \blacksquare \text{ drip} \\ \_ \text{ smooth} \\ \_ \text{ accuse} \\ \_ \text{ intercept} \\ \_ \text{ register} \\ \_ \text{ bill} \end{matrix} \qquad |\text{tap}\rangle = \begin{pmatrix} -0.05 + 0.71i \\ 0. + 0.j \\ -0. - 0.02i \\ -0. - 0.i \\ -0. + 0.i \\ 0.65 + 0.27i \\ -0. + 0.i \\ 0.02 - 0.01i \end{pmatrix} \begin{matrix} \blacksquare \text{ knock} \\ \_ \text{ carry} \\ \_ \text{ drip} \\ \_ \text{ smooth} \\ \_ \text{ accuse} \\ \blacksquare \text{ intercept} \\ \_ \text{ register} \\ \_ \text{ bill} \end{matrix}$$

We see that the amplitudes of the basis states that correspond to the correct word senses are high, while all other values are zero, neglecting small fluctuations from the imperfect fitting process. In conclusion, we denote that the word-sense disambiguation based on the superposition model works as intended, and experiments with larger datasets may be conducted on this basis.

4.2 Disambiguation based on density matrices

In this section we cover the results of the model presented in Section 3.2.

4.2.1 Fitting the model

Recall that we load the vector representation of the nouns through a QRAM method. Hence, the only quantum word representations we have to learn are those of the unambiguous sense verbs. The ambiguous verbs are later modeled as mixed states of their senses. The loss curve for the fitting process is given in Figure 4.9

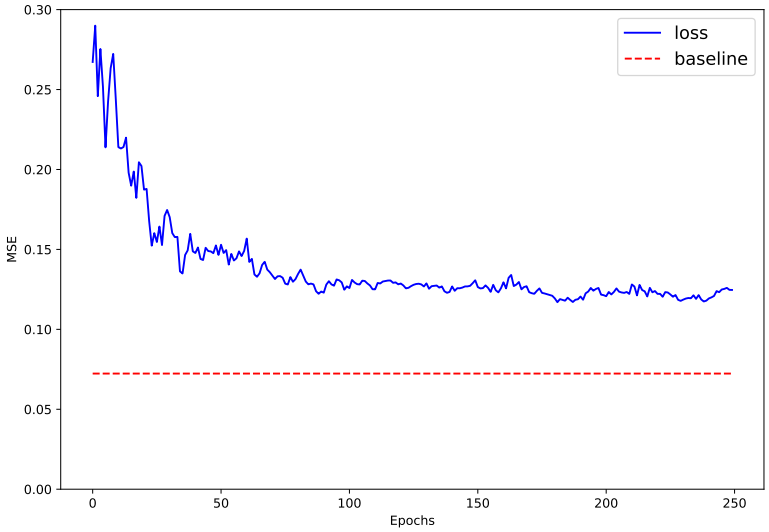


Figure 4.9: Loss curve for the fit of the unambiguous sense verbs. The blue curve depicts the MSE loss over the SPSA epochs, the red dashed line indicates the baseline determined by the linear model from Section 3.2.3.

The noun vectors are 4-dimensional data arrays. Thus, the model consists of two qubits. We use the 2-qubit strongly entangling layer as verb ansatz. The baseline analysis reveals a minimum possible mean-squared error of 0.0723. However, the model levels out at a loss of 0.119. The discrepancy between the quantum model and the baseline could be caused by the SPSA algorithm getting stuck in a local minimum or due to an unsuitable verb ansatz. However, testing the CNOT+U(3) ansatz did not improve the results. Although the fit is not perfect, we assume that it is sufficient for further analysis.

4.2.2 Constructing the density matrices

Extracting the state vectors of the verb ansätze lets us calculate the density matrices of the ambiguous verbs. In the example of "*file*", this is

$$\rho_{\text{file}} = \frac{1}{2}\left|\text{register}\right\rangle\left\langle\text{register}\right| + \frac{1}{2}\left|\text{smooth}\right\rangle\left\langle\text{smooth}\right| \tag{4.3}$$
$$= \begin{pmatrix} 0.439 & 0.052+0.112i & 0.024-0.034i & -0.152-0.079i \\ 0.052-0.112i & 0.446 & -0.128-0.059i & -0.019+0.048i \\ 0.024+0.034i & -0.128+0.059i & 0.046 & -0.01-0.019i \\ -0.152+0.079i & -0.019-0.048i & -0.01+0.019i & 0.069 \end{pmatrix}$$

The resulting density matrices are listed in Appendix A.3. To determine the level of uncertainty of the system described by the density matrix, we calculate the von Neumann entropy S for each ambiguous verb:

$$S(\rho_{\text{file}}) = 0.6554 \qquad S(\rho_{\text{dribble}}) = 0.6695$$
$$S(\rho_{\text{tap}}) = 0.6849 \qquad S(\rho_{\text{charge}}) = 0.6456.$$

Given that for a system living in a 4-D Hilbert space, the von-Neumann entropy is maximal $\ln(4) = 1.39$. Hence, we denote that the lexical ambiguity of the four verbs is fairly high. Composing the ambiguous verbs with their context words according to Equation 3.11, let us compute the disambiguation power. Furthermore, we compare the disambiguation power of the valid context words to those of random word vectors that have little similarity to the context words. The results are depicted in Figure 4.10.

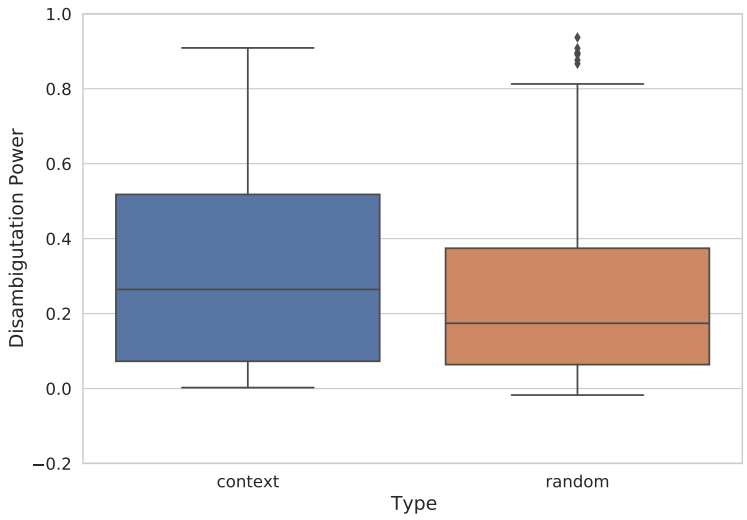


Figure 4.10: Disambiguation power of context words vs. random vectors. Context words are nouns that co-occur with the ambiguous word (e.g. "*application*" with "*file*"), whilst the control group consits of random vectors that share little similarity with all context words.

We record data from $n_{\text{context}} = 35$ real verb-context pairs and a random sample group of size $n_{\text{rand}} = 350$. The median disambiguation power of the context words is 0.31, while we record a value of 0.18 using random noun vectors. Testing the difference in

distribution using Wilcoxon rank-sum test yields a p value of $p = 0.298$. Hence, there is no proven effect of the disambiguation on the von Neuman entropy. Looking at the whiskers of the box plot, we denote that the disambiguation power, both for the context word group and the randomized control group, ranges within the complete spectrum of $[0, 1]$. We conclude that the measure of the disambiguation power using the von-Neumann entropy is not sufficient to show a successful disambiguation. Therefore, we continue to record the word similarities of the disambiguated verbs with their intended sense meaning according to Equation 3.16. Take the example of "*file*". We disambiguate ρ_{file} by composing it with the density matrix of the context verb "*account*":

$$\rho_{\text{file, account}} = \rho_{\text{file}} \odot |\text{account}\rangle \langle \text{account}| . \tag{4.4}$$

Now we calulate the similarity between the disambiguated densisty matrix $\rho_{\text{file, account}}$ and the abstract senses ρ_{register} and ρ_{smooth} through Equation 3.16. We then record the difference of the calculated similaries to the recentered ground truth values, such that a higher score eqals a better model:

$$\begin{array}{llll} \text{Low Similarity:} & s_1 = \text{Tr}(\rho^\dagger_{\text{smooth}}\ \rho_{\text{file, account}}) & \text{Recorded value:} & |1 - s_1| \\ \text{High Similarity:} & s_2 = \text{Tr}(\rho^\dagger_{\text{register}}\ \rho_{\text{file, account}}) & \text{Recorded value:} & |s_2| \end{array}$$

Furthermore, we calculate the similarity of randomly disambiguated words to the original senses. In other words, we calculate the composition

$$\rho_{\text{file, random}} = \rho_{\text{file}} \odot |\text{random}\rangle \langle \text{random}| \tag{4.5}$$

and calculate also the similarity to the abstract senses ρ_{register} and ρ_{smooth}. In an ideal model, those similaries should be low. The results are shown in Figure 4.11.

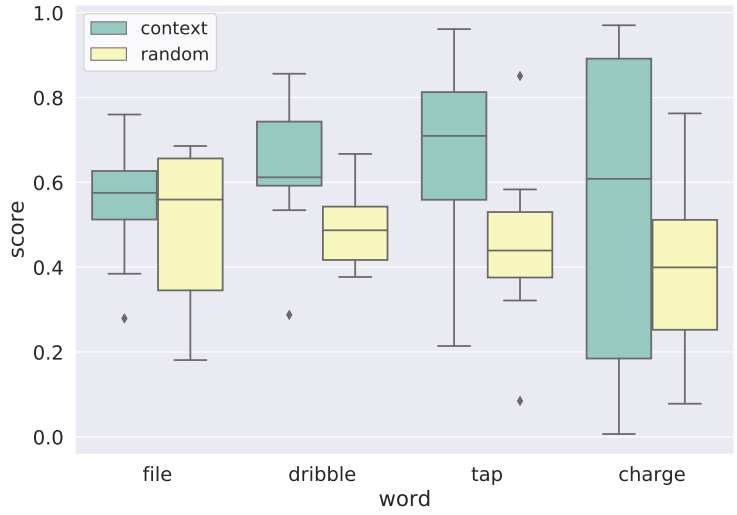


Figure 4.11: Similaries of the disambiguated verbs to their sense verbs in comparison to those disambiguated by randomly initialized density matrices and the supposed senses.

We test the difference in distribution for the similarities of the proper disambiguated verbs and those disambiguated through random density matrices by using Wilcoxon's rank-sum test. For the ambiguous verbs "*dribble*" and "*tap*" we record a p-value below 0.05 (see Table 4.1).

Verb	Wilcoxon p-value
file	0.217
dribble	< 0.01
tap	< 0.01
charge	0.942

Table 4.1: Wilcoxon's rank-sum test results after verb disambiguation.

Although failing for the verbs "*charge*" and "*file*", we see light evidence for successful disambiguation of ambiguous verbs by composing them with context nouns through the Hadamard multiplication. This result indicates a possible benefit of representing words as density matrices and performing a prior disambiguation. However, more sophisticated models for the prior word disambiguation have to be evaluated to confirm the validity of the density matrix formalism.

5
Conclusion

In this work, we examined the theory and practical application of the Compositional Distributional Model of Meaning applied to a word disambiguation task. We introduced the mathematical background of the model and discussed possible applications using quantum computers. Furthermore, we gave an overview of the CPM model, developed by Piedeleu et al. [27] and Kartsaklis [19], based on mixed quantum states. All calculations were carried out using a simulator for quantum circuits. The code is published on Github[1].

5.1 Discussion

For both the disambiguation models based on superposition (Section 3.1) and on density matrices (Section 3.2), we showed that encoding the meaning of ambiguous verbs into the proposed word embeddings yields meaningful results. We have seen a method of composing the ambiguous verb with a context noun to disambiguate it for both models. Statistically, we denote that encoding the ambiguous verb based on the superposition model yields more stable results. However, in a real-world scenario, encoding all possible verb senses of ambiguous words into a superposition of qubit states would require such a high-dimensional vector space that it is questionable whether learning a valid word representation is feasible. Therefore, the CPM model proposes a more natural way of encoding ambiguous word meanings. Here, we encode the standalone meanings of words (i.e., their senses) as vectors in a sufficiently large vector space. Words with multiple meanings are then encoded into density matrices of mixed quantum states, which carry more information than, for example, just a linear combination of vectors. However, one disadvantage of the CPM model is that the composition of words on a quantum circuit has not been carried out yet. Thus, we investigated a compositional model proposed by Kartsaklis [19], which uses the Hadamard product to compose density matrices of word meanings. Unfortunately, by conducting the experiment using classical operations, we lose the quantum advantage of the model. Although we see some statistical evidence for successful disambiguation (see Figure 4.11) using density matrices, the results have to be taken with a grain of salt. We can only show significant results for two of four ambiguous verbs, namely the verbs *dribble* and *tap*. Additionally, the underlying dataset has been developed exclusively for the task, and further real-world evaluations need to be conducted to verify or falsify the approach.

[1] https://github.com/Thommy257/Quantum-Models-for-Word-Sense-Disambiguation

5.2 Feasibility analysis

In general, it is not an easy task to prove the superiority of a quantum algorithm over its classical counterpart. However, because the Compositional Distributional Model of Meaning relies on the tensor product of vector spaces, we know that the memory usage on a classical computer would increase exponentially with the sentence length. At the same time, the quantum version only grows linearly in the number of qubits. Therefore, we know that, on a large scale, the model can only be computed efficiently on quantum devices. Nevertheless, criticisms of a purely quantum model are that quantum states are always l_2-normalized, i.e., even for a very high-dimensional Hilbert space, the state-space is highly limited. Hence, embedding word meanings into quantum states could be challenging. Furthermore, the error rate of the quantum operations can destroy any quantum advantage the model has over classical NLP approaches. With current NISQ-devices, only comparably small CDM models can be implemented ([24], [23]), and it is still unclear when fault-tolerant and scalable quantum computers will be available.
Another issue that might occur is related to the compositional structure of the model: Grammar rules are directly encoded into the model. However, before we implement them, we have to know them. For many decades, researchers have tried to identify a comprehensive set of rules for different languages [22], [21], [26], which happens to be unsolved, even for grammatically simple languages like English [34]. Moreover, we require a text parser that determines the word types in a sentence correctly. Such a parser may need additional resources for training and inference, which could undermine a quantum advantage even further.
Last but not least, the Compositional Distributional Model of Meaning appears to be well-studied for the English language; applying it to other languages is, on the other hand, not an easy task. Although some languages share some linguistic concepts in common, many rely on vastly different sets of grammatical rules. Hence, the model lacks the flexibility of state-of-the-art language models based on neural networks. Those networks infer the grammatical structure of a language from large corpora, making them versatile for different languages and a wide variety of NLP tasks.

5.3 Outlook and open questions

Further investigation is needed to implement a circuit model for the CPM model based on density matrices. The composition of words as mixed states through quantum operations like the "*fuzz*" and "*phaser*" operations introduced by Coecke and Meichanetzidis [10] shall be implemented and analyzed thoroughly. Furthermore, larger and more complex datasets (i.e., datasets with more cross-occurrences of senses and nouns) shall be used to fit the model to resemble real-life applications.
Applying the Compositional Distributional Model of Meaning on tasks like document retrieval could mark a significant leap forward for web searches. By embedding web pages into quantum states, we could benefit from the speed-up provided by Grover's algorithm to find matching pages for a given query.

Bibliography

[1] Héctor Abraham et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2019. DOI: 10.5281/zenodo.2562110.

[2] Samson Abramsky and Bob Coecke. "Categorical quantum mechanics". In: *Handbook of quantum logic and quantum structures* 2 (2009), pp. 261–325.

[3] Steve Awodey. *Category theory*. Oxford university press, 2010.

[4] Bernd Bohnet et al. "Morphosyntactic Tagging with a Meta-BiLSTM Model over Context Sensitive Token Encodings". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 2642–2652. DOI: 10.18653/v1/P18-1246. URL: https://www.aclweb.org/anthology/P18-1246.

[5] Tai-Danae Bradley. "What is Applied Category Theory?" In: *arXiv preprint arXiv:1809.05923* (2018).

[6] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[7] Stephen Clark. "Vector space models of lexical meaning". In: *The Handbook of Contemporary semantic theory* (2015), pp. 493–522.

[8] Bob Coecke and Ross Duncan. "Interacting quantum observables: categorical algebra and diagrammatics". In: *New Journal of Physics* 13.4 (2011), p. 043016.

[9] Bob Coecke and Aleks Kissinger. "Picturing quantum processes". In: *International Conference on Theory and Application of Diagrams*. Springer. 2018, pp. 28–31.

[10] Bob Coecke and Konstantinos Meichanetzidis. "Meaning updating of density matrices". In: *arXiv preprint arXiv:2001.00862* (2020).

[11] Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. "Mathematical foundations for a compositional distributional model of meaning". In: *arXiv preprint arXiv:1003.4394* (2010).

[12] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[13] Michael R Garey and David S Johnson. “Crossing number is NP-complete”. In: *SIAM Journal on Algebraic Discrete Methods* 4.3 (1983), pp. 312–316.

[14] Edward Grefenstette and Mehrnoosh Sadrzadeh. “Experimental Support for a Categorical Compositional Distributional Model of Meaning”. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, July 2011, pp. 1394–1404. URL: https://www.aclweb.org/anthology/D11-1129.

[15] Oxford Quantum Group. *DisCoPy Package (GitHub)*. https://github.com/oxford-quantum-group/discopy. 2020.

[16] Oxford Quantum Group. *QNLP experiment*. https://github.com/oxford-quantum-group/discopy/blob/ab2b356bd3cad1dfb55ca6606d6c4b4181fe590c/notebooks/qnlp-experiment.ipynb. Accessed: 2021-03-01. 2020.

[17] Benjamin Heinzerling and Michael Strube. “Sequence Tagging with Contextual and Non-Contextual Subword Representations: A Multilingual Evaluation”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 273–291. DOI: 10.18653/v1/P19-1027. URL: https://www.aclweb.org/anthology/P19-1027.

[18] Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. “Separating Disambiguation from Composition in Distributional Semantics”. In: *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 114–123. URL: https://www.aclweb.org/anthology/W13-3513.

[19] Dimitrios Kartsaklis. “Compositional distributional semantics with compact closed categories and Frobenius algebras”. PhD thesis. Oxford University, UK, 2014.

[20] Iordanis Kerenidis and Anupam Prakash. “Quantum Recommendation Systems”. In: *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Ed. by Christos H. Papadimitriou. Vol. 67. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 49:1–49:21. ISBN: 978-3-95977-029-3. DOI: 10.4230/LIPIcs.ITCS.2017.49. URL: http://drops.dagstuhl.de/opus/volltexte/2017/8154.

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

[22] Joachim Lambek. “Type grammar revisited”. In: *International conference on logical aspects of computational linguistics*. Springer. 1997, pp. 1–27.

[23] Robin Lorenz et al. *QNLP in Practice: Running Compositional Models of Meaning on a Quantum Computer*. 2021. arXiv: 2102.12846 [cs.CL].

[24] Konstantinos Meichanetzidis et al. “Quantum natural language processing on near-term quantum computers”. In: *arXiv preprint arXiv:2005.04147* (2020).

[25] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. URL: http://arxiv.org/abs/1301.3781.

[26] RICHARD MONTAGUE. "Universal grammar". In: *Theoria* 36.3 (1970), pp. 373–398. DOI: https://doi.org/10.1111/j.1755-2567.1970.tb00434.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1755-2567.1970.tb00434.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1755-2567.1970.tb00434.x.

[27] Robin Piedeleu et al. "Open System Categorical Quantum Semantics in Natural Language Processing". In: *6th Conference on Algebra and Coalgebra in Computer Science (CALCO 2015)*. Ed. by Lawrence S. Moss and Pawel Sobocinski. Vol. 35. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015, pp. 270–289. ISBN: 978-3-939897-84-2. DOI: 10.4230/LIPIcs.CALCO.2015.270. URL: http://drops.dagstuhl.de/opus/volltexte/2015/5539.

[28] Maria Schuld et al. "Circuit-centric quantum classifiers". In: *Physical Review A* 101.3 (Mar. 2020). ISSN: 2469-9934. DOI: 10.1103/physreva.101.032308. URL: http://dx.doi.org/10.1103/PhysRevA.101.032308.

[29] Peter Selinger. "Dagger Compact Closed Categories and Completely Positive Maps: (Extended Abstract)". In: *Electronic Notes in Theoretical Computer Science* 170 (2007). Proceedings of the 3rd International Workshop on Quantum Programming Languages (QPL 2005), pp. 139–163. ISSN: 1571-0661. DOI: https://doi.org/10.1016/j.entcs.2006.12.018. URL: https://www.sciencedirect.com/science/article/pii/S1571066107000606.

[30] Priya Shree. *The Journey of Open AI GPT models*. https://medium.com/walmartglobaltech/the-journey-of-open-ai-gpt-models-32d95b7b7fb2. Accessed: 2021-03-31. 2020.

[31] Seyon Sivarajah et al. "t|ket⟩: a retargetable compiler for NISQ devices". In: *Quantum Science and Technology* 6.1 (2020), p. 014003.

[32] Ryan Mohammad Solgi. *Geneticalgorithm Python library*. https://github.com/rmsolgi/geneticalgorithm. 2020.

[33] J.C. Spall. "Implementation of the simultaneous perturbation algorithm for stochastic optimization". In: *IEEE Transactions on Aerospace and Electronic Systems* 34.3 (1998), pp. 817–823. DOI: 10.1109/7.705889.

[34] Bernhard Waltl, Georg Bonczek, and Florian Matthes. "Rule-based information extraction: Advantages, limitations, and perspectives". In: *Jusletter IT (02 2018)* (2018).

[35] Xanadu. *Pennylane Strongly Entangling Layers*. https://pennylane.readthedocs.io/en/stable/code/api/pennylane.templates.layers.StronglyEntanglingLayers.html. 2020.

[36] Ikuya Yamada et al. "Wikipedia2Vec: An Efficient Toolkit for Learning and Visualizing the Embeddings of Words and Entities from Wikipedia". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2020, pp. 23–30.

[37] William Zeng and Bob Coecke. “Quantum algorithms for compositional natural language processing”. In: *arXiv preprint arXiv:1608.01406* (2016).

A

Appendix

A.1 Full dataset

sentence	score
file account	1
register account	1
smooth account	0
file nail	1
register nail	0
smooth nail	1
file charge_n	1
register charge_n	1
smooth charge_n	0
file tooth	1
register tooth	0
smooth tooth	1
file application	1
register application	1
smooth application	0
file steel	1
register steel	0
smooth steel	1
file lawsuit	1
register lawsuit	1
smooth lawsuit	0
file shank	1
register shank	0
smooth shank	1
file declaration	1
register declaration	1
smooth declaration	0
charge rate	1
bill rate	1
accuse rate	0
charge person	1
bill person	0
accuse person	1
charge price	1
bill price	1
accuse price	0

sentence	score
charge demonstrator	1
bill demonstrator	0
accuse demonstrator	1
charge rent	1
bill rent	1
accuse rent	0
charge defendant	1
bill defendant	0
accuse defendant	1
charge interest	1
bill interest	1
accuse interest	0
charge fee	1
bill fee	1
accuse fee	0
charge offender	1
bill offender	0
accuse offender	1
charge commission	1
bill commission	1
accuse commission	0
tap table	1
intercept table	0
knock table	1
tap telephone	1
intercept telephone	1
knock telephone	0
tap tree	1
intercept tree	0
knock tree	1
tap conversation	1
intercept conversation	1
knock conversation	0
tap resource	1
intercept resource	1
knock resource	0

sentence	score
tap pencil	1
intercept pencil	0
knock pencil	1
tap door	1
intercept door	0
knock door	1
tap network	1
intercept network	1
knock network	0
tap market	1
intercept market	1
knock market	0
tap floor	1
intercept floor	0
knock floor	1
dribble ball	1
carry ball	1
drip ball	0
dribble wine	1
carry wine	0
drip wine	1
dribble blood	1
carry blood	0
drip blood	1
dribble melon	1
carry melon	0
drip melon	1
dribble tea	1
carry tea	0
drip tea	1
dribble sphere	1
carry sphere	1
drip sphere	0

A.2 Noun vectors

account	[-0.915559	-0.290692	-0.023149	0.276973]
application	[-0.628319	-0.341554	-0.677950	-0.170117]
ball	[0.744622	0.504402	0.344781	-0.268779]
blood	[0.639304	0.621144	0.240529	0.384209]
charge_n	[-0.892284	0.298885	0.137742	-0.309069]
commission	[-0.653919	-0.378497	-0.639725	0.141004]
conversation	[-0.399987	0.007231	-0.828033	0.392835]
declaration	[-0.388970	-0.280224	-0.709476	0.516547]
defendant	[-0.662875	0.646758	0.059432	0.372516]
demonstrator	[-0.111074	0.260220	-0.460983	-0.841096]
door	[0.728066	0.090073	-0.097129	-0.672587]
fee	[-0.675151	0.039193	0.736544	0.011742]
floor	[0.531391	-0.188479	-0.093965	-0.820530]
interest	[-0.842534	-0.473995	0.255117	0.019525]
lawsuit	[-0.766359	0.395978	-0.059764	0.502318]
market	[-0.073115	-0.933648	0.315943	-0.152103]
melon	[0.884770	-0.233734	0.059027	0.398831]
nail	[0.697880	0.702173	-0.113680	0.083622]
network	[-0.334869	-0.585165	-0.640153	-0.368305]
offender	[-0.690536	0.686349	0.221559	0.054736]
pencil	[0.764767	0.381652	-0.516504	-0.051939]
person	[-0.724748	0.649681	-0.228809	0.017373]
price	[-0.114293	-0.098936	0.985751	-0.073780]
rate	[-0.399495	-0.419783	0.625650	-0.522253]
rent	[-0.358408	-0.089284	0.926873	-0.066926]
resource	[-0.544162	-0.759249	-0.347160	-0.083120]
shank	[0.674325	0.710080	0.135126	-0.151040]
sphere	[0.059612	-0.374176	-0.766280	-0.518897]
steel	[0.820411	0.510913	-0.256681	-0.002872]
table	[0.707534	-0.700584	0.054560	0.074840]
tea	[0.635152	-0.542467	0.230359	0.499245]
telephone	[-0.407338	-0.815928	-0.410218	0.007694]
tooth	[0.669310	0.721324	-0.039959	-0.173547]
tree	[0.954451	-0.223417	-0.121512	-0.156021]
wine	[0.529559	-0.294460	0.329419	0.724115]

A.3 Density matrices

$\rho_{\text{file}} =$

$$\begin{bmatrix} 0.4387 + 0.i & 0.0521 - 0.1117i & 0.0244 + 0.0339i & -0.1524 + 0.0791i \\ 0.0521 + 0.1117i & 0.4463 + 0.i & -0.1279 + 0.0594i & -0.0189 - 0.0476i \\ 0.0244 - 0.0339i & -0.1279 - 0.0594i & 0.0461 + 0.i & -0.0103 + 0.0193i \\ -0.1524 - 0.0791i & -0.0189 + 0.0476i & -0.0103 - 0.0193i & 0.0689 + 0.i \end{bmatrix}$$

$\rho_{\text{dribble}} =$

$$\begin{bmatrix} 0.4357 + 0.i & 0.0673 + 0.0108i & -0.1232 - 0.0402i & 0.1323 - 0.0738i \\ 0.0673 - 0.0108i & 0.0894 + 0.i & -0.0756 - 0.0733i & -0.0839 - 0.1228i \\ -0.1232 + 0.0402i & -0.0756 + 0.0733i & 0.1403 + 0.i & 0.1381 + 0.0182i \\ 0.1323 + 0.0738i & -0.0839 + 0.1228i & 0.1381 - 0.0182i & 0.3346 + 0.i \end{bmatrix}$$

$\rho_{\text{tap}} =$

$$\begin{bmatrix} 0.4555 + 0.i & 0.0599 + 0.0134i & 0.0984 - 0.0269i & -0.1325 + 0.0775i \\ 0.0599 - 0.0134i & 0.3297 + 0.i & 0.1851 - 0.015i & 0.1045 - 0.0129i \\ 0.0984 + 0.0269i & 0.1851 + 0.015i & 0.1161 + 0.i & 0.0319 - 0.0024i \\ -0.1325 - 0.0775i & 0.1045 + 0.0129i & 0.0319 + 0.0024i & 0.0986 + 0.i \end{bmatrix}$$

$\rho_{\text{charge}} =$

$$\begin{bmatrix} 0.2162 + 0.i & -0.2205 - 0.0354i & 0.0012 - 0.0824i & 0.0042 + 0.0756i \\ -0.2205 + 0.0354i & 0.2318 + 0.i & 0.0325 + 0.0791i & -0.0229 - 0.071i \\ 0.0012 + 0.0824i & 0.0325 - 0.0791i & 0.4591 + 0.i & -0.1784 + 0.0793i \\ 0.0042 - 0.0756i & -0.0229 + 0.071i & -0.1784 - 0.0793i & 0.0929 + 0.i \end{bmatrix}$$

A.4 Initializing mixed states on a quantum circuit

A mixed quantum state is fundamentally different from a pure quantum state. However, in terms of quantum circuits, we are usually dealing with well-defined pure quantum states. To represent mixed states, we need to apply some tricks. One possibility is to represent the statistical ensemble of pure states by initializing each state probabilistically. However, this requires us to run the algorithm many times and sample the results. Another possibility is to entangle the states of the ensemble to ancilla qubits and discarding them afterward.

Procedure Consider two words $|\text{word}_1\rangle$ and $|\text{word}_2\rangle$, which are represented as states of a multi-qubit system. Suppose that these words encode the senses of an ambiguous word with equal probabilities, we can write the density matrix of such a word as

$$\rho = \frac{1}{2} |\text{word}_1\rangle \langle\text{word}_1| + \frac{1}{2} |\text{word}_2\rangle \langle\text{word}_2| \tag{A.1}$$

To represent the mixed state in a circuit, we first initialize the state

$$|\psi\rangle_{W\otimes A} = \frac{1}{\sqrt{2}} \left(|\text{word}_1\rangle_W |0\rangle_A + |\text{word}_2\rangle_W |1\rangle_A\right) \tag{A.2}$$

where the words live in a qubit register denoted by W and the ancilla qubit(s) in register A. Such a state can be initialized by the circuit shown in Figure A.1.

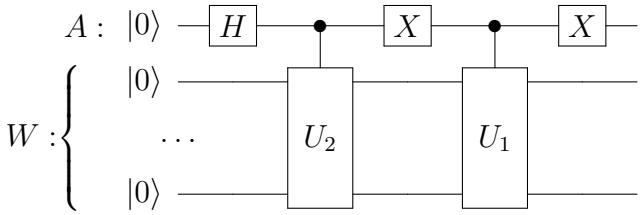


Figure A.1: Quantum circuit to initialize the state shown in Equation A.2. U_1 and U_2 act as state preparation unitaries s.t. $U_i : |0\rangle^{\otimes |W|} \to |\text{word}_i\rangle$.

Evidently, the density matrix of the purified state $|\psi\rangle_{W\otimes A}$ follows:

$$\begin{aligned}
\rho^{WA} = |\psi\rangle \langle\psi|_{W\otimes A} &= \frac{1}{2} \left(|\text{word}_1\rangle_W |0\rangle_A + |\text{word}_2\rangle_W |1\rangle_A\right) \\
&\quad \cdot \left(\langle\text{word}_1|_W \langle 0|_A + \langle\text{word}_2|_W \langle 1|_A\right) \\
&= \frac{1}{2}((|\text{word}_1\rangle \langle\text{word}_1|_W \otimes |0\rangle \langle 0|_A) \\
&\quad + (|\text{word}_1\rangle \langle\text{word}_2|_W \otimes |0\rangle \langle 1|_A) \\
&\quad + (|\text{word}_2\rangle \langle\text{word}_1|_W \otimes |1\rangle \langle 0|_A) \\
&\quad + (|\text{word}_2\rangle \langle\text{word}_2|_W \otimes |1\rangle \langle 1|_A)).
\end{aligned}$$

Tracing out (i.e. discarding) the ancilla A then leaves register W in the mixed state ρ as defined in Equation A.1:

$$\begin{aligned}\mathrm{Tr}_A(\rho^{WA}) = \frac{1}{2}(&\mathrm{Tr}_A\left(|\mathrm{word}_1\rangle\langle\mathrm{word}_1|_W \otimes |0\rangle\langle 0|_A\right)\\ &+ \mathrm{Tr}_A\left(|\mathrm{word}_1\rangle\langle\mathrm{word}_2|_W \otimes |0\rangle\langle 1|_A\right)\\ &+ \mathrm{Tr}_A\left(|\mathrm{word}_2\rangle\langle\mathrm{word}_1|_W \otimes |1\rangle\langle 0|_A\right)\\ &+ \mathrm{Tr}_A\left(|\mathrm{word}_2\rangle\langle\mathrm{word}_2|_W \otimes |1\rangle\langle 1|_A\right))\\ = \frac{1}{2}&|\mathrm{word}_1\rangle\langle\mathrm{word}_1|_W + \frac{1}{2}|\mathrm{word}_2\rangle\langle\mathrm{word}_2|_W .\end{aligned}$$

To initialize other mixtures, one could increase the number of ancilla qubits and replace the Hadamard gate in Figure A.1 with a parametrized R_y rotation.

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY