

Saskia Bruhn

Density Matrix Methods in Quantum Natural Language Processing

PICS 2022, Number 2

PICS

Publications of the Institute of Cognitive Science

Editorial board:

Annette Hohenberger
Simone Pika
Gordon Pipa
Achim Stephan
Tobias Thelen

Saskia Bruhn. 2022. *Density Matrix Methods in Quantum Natural Language Processing* (Publications of the Institute of Cognitive Science 2022, Number 2). Osnabrück: Institute of Cognitive Science, Osnabrück University.

This title can be downloaded at:
<https://osnads.uosnabrueck.de>

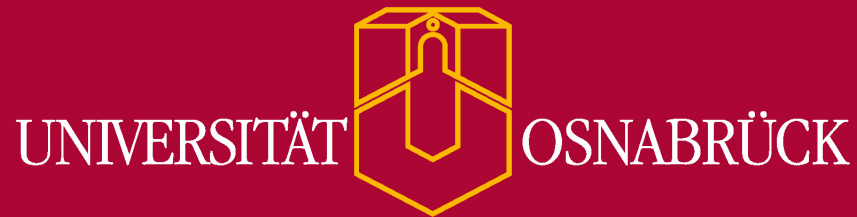
© 2022 Saskia Bruhn

Published under the Creative Commons Attribution 3.0 Germany:
<http://creativecommons.org/licenses/by/3.0/de/>

Institut für Kognitionswissenschaft
Universität Osnabrück
49069 Osnabrück
Germany
<https://ikw.uni-osnabrueck.de>

Storage and cataloging done by Osnabrück University





UNIVERSITY OF OSNABRÜCK
INSTITUTE OF COGNITIVE SCIENCE

Density Matrix Methods in Quantum Natural Language Processing

Master's Thesis

submitted by

Saskia Bruhn

First Examiner: Dr. Mark Mattingley-Scott
Second Examiner: Prof. Dr. Kai-Uwe Kühnberger
Supervisor: Dr. Rukhsan Ul Haq

Osnabrück, December 19, 2021

Abstract

Though vectors are the most commonly used structure to encode the meaning of words computationally, they fail to represent uncertainty about the underlying meaning. Ambiguous words can be best described by probability distributions over their various possible meanings. Putting them in context should disambiguate their meaning. Similarly, lexical entailment relationships can be characterized using probability distributions. A word higher up in the hierarchical order is then modeled as a probability distribution over the meanings of words it subsumes. The DisCoCat model, which is inspired by the mathematical structure of quantum theory, proposes density matrices as word embeddings that are able to capture this structure. In quantum mechanics, they describe systems whose states are only known with uncertainty. First experiments have proven their ability to capture word similarity, word ambiguity, and lexical entailment structures. An adaption of the Word2Vec model, called Word2DM, can learn such density matrix word embeddings. To enforce that the learned matrices possess the properties of density matrices, the model learns intermediary matrices and derives the density matrices from them. This strategy causes the parameter updates to be sub-optimal. This thesis proposes a hybrid quantum-classical algorithm for learning density matrix word embeddings to resolve this issue. Exploiting the fact that density matrices naturally describe quantum systems, no intermediary matrices are needed, and the shortcomings of the classical Word2DM model can theoretically be circumvented. The parameters of a variational quantum circuit are optimized such that the qubits' state corresponds to the word's meaning. The state's density matrix description is then extracted and used as word embedding. A separate set of parameters corresponding to its density matrix embedding is learned for each word in the vocabulary. A first implementation has been executed on a quantum simulator in the course of this thesis. The utilized objective function decreases the distance between co-occurring words and increases the distance between words that do not occur together. The training success can therefore be measured by evaluating the similarity of the learned word embeddings. The model was trained on text corpora with small vocabulary sizes. The learned embeddings showed the expected similarities between the words in the text. Implementation issues on real quantum hardware like extracting complete state representations and calculating gradients for this model will also be discussed.

Acknowledgements

First of all, I would like to thank Prof. Dr. Kai-Uwe Kühnberger for always being willing to help and advise his students, for connecting me with Dr. Mark Mattingley-Scott, which made this master's thesis possible, and for acting as the second examiner.

I would like to thank Dr. Mark Mattingley-Scott for giving me the opportunity to write this thesis, for introducing me to the research field of Quantum Natural Language Processing, and for acting as the first examiner of this thesis.

Thank you to my supervisor Dr. Rukhsan Ul Haq, for helping me orientate myself in this research field and helping me identify the topic for this thesis. I appreciate your enthusiasm and continued support.

Thank you to Dr. Martha Lewis, Francois Meyer, and Anton Dekusar for taking the time to answer my questions.

I would also like to thank Dr. Florian Preis, Inga, and Benedict for the final proofreading. Thank you to Inga for her support and encouragement and the fact that I can always rely on her.

Last, but not least, my warm and heartfelt thanks go to my parents and my brother for their tremendous love and support and for always believing in me.

Contents

1	Introduction	1
1.1	Two Paradigms in Natural Language Processing	1
1.2	A Quantum-Inspired Model of Language	1
1.3	Modeling Word Ambiguity and Lexical Entailment with Density Matrices	2
1.4	Quantum Density Matrix Word Embeddings	2
1.5	Thesis Structure	3
2	Quantum Computation	5
2.1	Qubits	5
2.2	Measurement	6
2.3	The Bloch Sphere	7
2.4	Multiple-Qubit Systems	9
2.5	Entanglement	10
2.6	Pure States, Mixed States, and Density Operator	11
2.7	Quantum Circuits	13
2.8	Variational Quantum Circuits	16
2.8.1	Gradients	17
3	Distributional Compositional Model of Meaning	19
3.1	Category-Theoretic Background	20
3.1.1	Graphical Calculus	22
3.2	Vector Space Model of Meaning	25
3.3	Lambek’s Pregroup Grammar	26
3.4	Category-Theoretic Combination of Grammar and Meaning	28
3.4.1	Vector Spaces as Compact Closed Categories	28
3.4.2	Pregroups as Compact Closed Categories	28
3.4.3	Sentence Meanings	29
3.5	Going Quantum	31
3.5.1	Quantum Advantage	33

4	Word Meaning with Density Matrices	35
4.1	Density Matrices as Compact Closed Category	37
4.2	Meaning of Sentences in CPM(FHilb)	40
4.3	Word Similarity with Density Matrices	42
4.3.1	Trace Inner Product	42
4.3.2	Trace Distance	43
4.3.3	Fidelity	43
4.4	Lexical Entailment with Density Matrices	44
4.5	Density Matrices for Word Ambiguity	49
4.6	Meaning Updating of Density Matrices	51
4.7	Learning Density Matrices Classically	53
4.7.1	Word2Vec Continuous Skip-gram	54
4.7.2	Word2DM	57
5	Quantum Density Matrix Word Embeddings	59
5.1	Number of Qubits	60
5.2	Choice of Circuit Ansatz	60
5.3	Extraction of Density Matrices from the Circuit	61
5.3.1	Quantum State Tomography	61
5.3.2	Simulators	61
5.4	Gradients	62
5.5	Implementation	62
5.5.1	Choice of Package	64
5.6	Evaluation - Choice of Similarity Measure	65
5.7	Data	65
5.7.1	ABC Dataset	65
5.7.2	Animal Dataset	66
6	Results and Discussion	67
6.1	ABC Dataset	67
6.1.1	Single BasicEntanglerLayer	67
6.1.2	Two BasicEntanglerLayers	68
6.2	Animal Dataset	69
7	Summary and Further Work	71
7.1	Summary	71

7.2	Further Work	72
7.2.1	Circuit Design	72
7.2.2	Similarity Measure and Objective Function	72
7.2.3	Word Ambiguity and Lexical Entailment	72
7.2.4	Implementation on Real Quantum Hardware	73
	List of Figures	75
	List of Tables	77
	Bibliography	79

1

Introduction

1.1 TWO PARADIGMS IN NATURAL LANGUAGE PROCESSING

Natural language processing is a subfield of artificial intelligence that enables computers to understand the content of written language and human speech. Therefore, different aspects of natural language have to be incorporated, such as the meaning of words and the grammatical structure of sentences. Two different modeling paradigms can be distinguished: Distributional and compositional methods.

Distributional methods learn the meaning of words based on the contexts the words appear in. Usually, word meanings are encoded into vectors and trained to be more similar to words in whose context they occur and less similar to those they do not occur with [1, 2]. The grammatical structure of sentences can be formalized in compositional approaches based on the composition of the words' grammatical types [3].

Neither distributional nor compositional methods can capture both these aspects of language at once. Furthermore, techniques that combine distributional and compositional methods to compute the meaning of whole sentences face problems, such as the fact that the meaning representations of two sentences with different grammatical structures live in different vector spaces and thus cannot be compared [4].

1.2 A QUANTUM-INSPIRED MODEL OF LANGUAGE

In [5], Coecke et. al. introduced a distributional compositional model of language (DisCoCat model) that operates in the framework of compact closed monoidal categories. It is able to reflect both the meaning of words as well as the grammatical structure of sentences and overcomes the issues of [4]. It depicts the meaning of words in a sentence as vectors and composes them using the tensor product, while the grammatical structure is encoded into a linear map. Applying this linear map to the tensor product of word meaning vectors yields a vector that represents the meaning of the entire sentence. Since this approach is based on the tensor product, the vector spaces in which it operates become very high-dimensional. The capacity of classical computing power is therefore exceeded for large text corpora [6]. The model was indeed inspired by quantum protocols such as quantum teleportation [7] in

the formulation of categorical quantum mechanics [8]. The mathematical similarity allows translating the computations within the DisCoCat model into quantum programs, called quantum circuits, and carrying them out on real quantum hardware [9, 10]. The meaning of words are then quantum states, and quantum operations on these states represent the grammatical structure. Since quantum systems are naturally composed using the tensor product, this reduces the number of necessary qubits enormously compared to classical bits [6].

1.3 MODELING WORD AMBIGUITY AND LEXICAL ENTAILMENT WITH DENSITY MATRICES

The meaning of ambiguous words and hyperonyms that subsume multiple concepts can be best described by probability distributions over their underlying word senses. Ambiguous words like ‘bank’ are then modeled as probability distribution over their possible senses: $bank = 1/3 \textit{ financial institute} + 1/3 \textit{ sand bank} + 1/3 \textit{ river bank}$. Similarly, hyperonyms like ‘mammal’ are modeled as a probability distribution over the words that they subsume: $mammal = 1/3 \textit{ dog} + 1/3 \textit{ dolphin} + 1/3 \textit{ human}$.

While vectors fail to represent these phenomena, quantum states provide a solution. There are different types of quantum states, pure and mixed ones. Pure states describe systems whose state is known with certainty and can be depicted as vectors. States that are only known with uncertainty are called mixed states and are represented by density matrices. These reflect a probability distribution over possible pure states a quantum system might be in. The use of density matrices to model word ambiguity and lexical entailment was proposed in [11] and [12].

In [13], the widely used Word2Vec model [2] that learns meaning vectors from text corpora based on their context was adapted to learn density matrices. This model is called the Word2DM model. To ensure that the learned matrices have the properties of density matrices, some intermediary matrices are learned from which the actual word embedding density matrices are derived. Unfortunately, this strategy introduces a lot of additional matrix multiplications into the objective function and, in some cases, causes the gradients to become so small that the parameters cannot be updated effectively.

1.4 QUANTUM DENSITY MATRIX WORD EMBEDDINGS

This thesis proposes a hybrid quantum-classical model to learn density matrix word embeddings using parameterized quantum circuits, referred to as variational quantum circuits.

Since density matrices naturally describe quantum states, the problems of the Word2DM model are circumvented, and no intermediary matrices are needed. For each word in the vocabulary, a separate parameter set is learned that prepares the system into a state that corresponds to the word's meaning. Extracting the state's density matrix description provides the word embedding. For training purposes, the objective function of the Word2DM model is utilized. It aims to reduce the distance between co-occurring words and increase the distance between words that do not occur together. Thus, the training success can be measured by evaluating the similarity of the learned word embeddings.

1.5 THESIS STRUCTURE

The structure of this thesis follows the storyline of this introduction. Chapter two introduces the basic concepts of quantum theory, quantum computation, and variational quantum circuits.

The third chapter explains the DisCoCat model [4]. First, the category-theoretical background is outlined, followed by the vector space model of meaning and Lambek's pregroup grammar. The unification of these three components into one distributional compositional model for sentence meanings is then demonstrated. The final section of chapter three presents the implementation of the DisCoCat model on quantum hardware and the advantage to be gained.

Chapter four first shows how density matrices can be integrated into the DisCoCat model. Then, different similarity measures for density matrices are introduced and the ability of density matrices to capture word ambiguity and lexical entailment is demonstrated. Two composition methods for density matrices are additionally introduced as an alternative to the tensor product. Finally, the widely used Word2Vec model and its adaption for learning density matrix word embeddings, the Word2DM model, are explained.

The fifth chapter presents the achievement of this thesis: a hybrid quantum-classical algorithm that learns quantum density matrix word embeddings. Architecture design choices, the extraction of density matrices from quantum circuits, and the gradient computation are discussed. Finally, the first implementation using a simulator is demonstrated.

Chapter 6 evaluates the learned density matrix word embeddings and discusses the results. The last chapter summarizes this thesis and gives an outlook on further work and possible improvements of this model.

2 | Quantum Computation

This chapter introduces the fundamental concepts of quantum computation following the standard textbooks ‘Quantum Computation and Quantum Information’ [14], ‘Quantum Machine Learning’ [15] and the Qiskit online textbook [16]. While it is not intended to provide a complete introduction to quantum mechanics and quantum computing, but rather a selected overview and comprehension of the principles relevant to this thesis, a more extensive description can be found in the mentioned resources above.

Initially, the concepts of quantum bits and quantum states are introduced. Next, the quantum mechanical rules of measurement, entanglement, and mixed quantum states are explained using the example of qubit systems. Ultimately, an overview of quantum gates and quantum circuits provides the necessary foundations to understand the inner workings of the quantum density matrix word embedding model presented later in this work.

2.1 QUBITS

Classical computation is based on the manipulation of bits. They can either take the value 0 or 1. In quantum computation, quantum bits are the analogous concept to bits, also called qubits. Like the classical bit, a qubit can also assume the two states $|0\rangle$ and $|1\rangle$. The notation ‘ $|\dots\rangle$ ’ originates from the Dirac notation in quantum mechanics, otherwise known as Bra-Ket notation. Vectors are written in these closing brackets, and their duals are written in opening brackets $\langle\dots|$. Hence, the notation for dual vectors is also referred to as ‘Bra’, and that for vectors is called ‘Ket’. The states $|0\rangle$ and $|1\rangle$ are the basis vectors of a two-dimensional complex Hilbert space and can also be described as follows:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2.1}$$

The decisive feature of qubits is the circumstance by which they are not limited to just one or the other of two states but can take on a full spectrum of states between $|0\rangle$ and $|1\rangle$. A qubit in such a state is considered to be in superposition. Mathematically, a superposition

state $|\psi\rangle$ is expressed as a linear combination of the basis states.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad \text{with } \alpha, \beta \in \mathbb{C} \quad (2.2)$$

Where α and β are the amplitudes of the respective basis states. Since the state of a qubit is always normalized, $|\alpha|^2 + |\beta|^2 = 1$ must be fulfilled. The following state is an example for a state in superposition:

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

2.2 MEASUREMENT

While observing a state in one of the basis states is deterministic, the state of a qubit in superposition is generally not fully observable. This marks another difference to classical bits. When measuring the state of a qubit, only one of the basis states involved in the superposition is observable. The respective basis states are measured with the probability described by their amplitudes.

$$|\langle i|\psi\rangle|^2 = |\alpha_i|^2, \quad \text{with } i \in \{0, 1\} \text{ and } \alpha_i \in \{\alpha, \beta\} \quad (2.3)$$

Example 1. Measuring the state

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

results in the observation of state $|0\rangle$ with the following probability:

$$|\langle 0|\psi\rangle|^2 = \left| \frac{1}{\sqrt{2}}\langle 0|0\rangle + \frac{1}{\sqrt{2}}\langle 0|1\rangle \right|^2 = \left| \frac{1}{\sqrt{2}}\langle 0|0\rangle \right|^2 = \frac{1}{2}$$

Analogously the state $|1\rangle$ is also observed with probability $\frac{1}{2}$.

A repeated measurement after a first measurement always yields the same result as the first one, i.e., the first measurement changes the qubit's state irreversibly into the measured one. Thus, if the first measurement returned the state $|0\rangle$ in the above example, the system would no longer be in the superposition state, but in state $|0\rangle$.

2.3 THE BLOCH SPHERE

Single-qubit states can also be expressed in a different form that allows illustrating them as vectors pointing to the surface of a unit sphere. This kind of illustration is the so-called Bloch Sphere. A quantum state is then described in the following form:

$$\begin{aligned} |\psi\rangle &= e^{i\phi_1}\alpha|0\rangle + e^{i\phi_2}\beta|1\rangle, & \text{with } \alpha, \beta \in \mathbb{R} \\ &= e^{i\phi_1}(\alpha|0\rangle + e^{i(\phi_2-\phi_1)}\beta|1\rangle) \end{aligned} \quad (2.4)$$

Due to the definition of the absolute square of complex numbers $|z|^2 = (re^{i\phi})(r^*e^{-i\phi})$, the factor $e^{i\phi_1}$ vanishes when calculating $|\langle\psi|\psi\rangle|^2$. Thus, the value of ϕ_1 is not observable. The parameters in equation 2.4 are renamed as $\gamma := \phi_1$ and $\phi = \phi_2 - \phi_1$. γ is the co-called global phase of the state and ϕ is the so-called relative phase. Since the global phase is not observable, states with different global phases are equivalent.

$$|\psi\rangle = e^{i\gamma}|\psi\rangle \quad (2.5)$$

Equation 2.4 can therefore be written as follows:

$$|\psi\rangle = \alpha|0\rangle + e^{i\phi}\beta|1\rangle, \quad \text{with } \alpha, \beta \in \mathbb{R} \quad (2.6)$$

Due to the normalisation of the state $\sqrt{\alpha^2 + \beta^2} = 1$ and the trigonometric identity $\sqrt{\sin^2 x + \cos^2 x} = 1$, α and β can also be expressed in terms of one variable θ :

$$\alpha = \cos\frac{\theta}{2}, \quad \beta = \sin\frac{\theta}{2}$$

Thus, single-qubit states are fully described by the following equation:

$$|\psi\rangle = e^{i\gamma} \left(\cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle \right), \quad \text{with } 0 \leq \theta \leq \pi \text{ and } 0 \leq \phi \leq 2\pi \quad (2.7)$$

As seen in equation 2.5, $e^{i\gamma}$ can be neglected and the expression reduces to:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle \quad (2.8)$$

This term for quantum states describes all possible vectors pointing to the surface of a three-dimensional unit sphere. θ describes the angle by which the respective vector and the Z-axis intersect, while ϕ describes the angle by which the vector and the X-axis intersect. Figure 2.1 shows an illustration of a single-qubit state in the Bloch sphere.

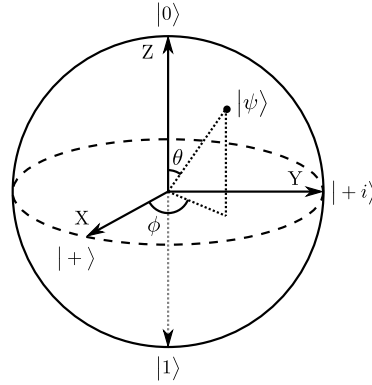


Figure 2.1: Illustration of the Bloch sphere representation of a single-qubit state ψ pointing to the sphere's surface. The angle ϕ describes the angle between the state's vector and the X-axis. Consequently, θ describes the angle between the state and the Z-axis. In addition to their designation, the respective basis states are also marked on the axes.

Example 2. State $|0\rangle$ is the vector on the positive Z-axis for which both angles in equation 2.8 are zero:

$$|0\rangle = \cos\frac{0}{2}|0\rangle + e^{i\cdot 0}\sin\frac{0}{2}|1\rangle$$

State $|1\rangle$ is the vector on the negative Z-axis for which $\theta = \pi$ and $\gamma = 0$:

$$|1\rangle = \cos\frac{\pi}{2}|0\rangle + e^{i\cdot 0}\sin\frac{\pi}{2}|1\rangle$$

Since the two states $|0\rangle$ and $|1\rangle$ form a basis of the two-dimensional complex Hilbert space, they are called the Z-basis. The two vectors on the positive and negative X-axis form the X-basis and are noted as $|+\rangle$ and $|-\rangle$. Accordingly, the vectors on the positive and negative Y-axis form the Y-basis $|+i\rangle$ and $|-i\rangle$. Any state can be expressed in each of the three bases. For example, the basis states of the X-basis can be written in the Z-basis as follows:

$$\begin{aligned} |+\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ |-\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned} \tag{2.9}$$

Correspondingly, the basis states of the Y-basis can be written in the Z-basis as follows:

$$\begin{aligned} | + i \rangle &= \frac{1}{\sqrt{2}} \left(|0\rangle + i|1\rangle \right) \\ | - i \rangle &= \frac{1}{\sqrt{2}} \left(|0\rangle - i|1\rangle \right) \end{aligned} \quad (2.10)$$

2.4 MULTIPLE-QUBIT SYSTEMS

Classical bits can be composed in systems of multiple bits. For example, a system of two bits can be in either one of the four states 00, 01, 10, and 11. Analogously, qubits can also be composed in systems of multiple qubits and a two-qubit system can be in one of the four states $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$. These form the basis of a four-dimensional complex Hilbert space. Just like single qubits, multiple qubits can also be in a superposition of the basis states:

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \quad (2.11)$$

More generally, the state of a n-qubit system is described by the tensor product of the individual states of the n qubits:

$$\begin{aligned} |\psi\rangle &= |\psi_1\rangle \otimes \dots \otimes |\psi_n\rangle \\ &= |\psi_1 \dots \psi_n\rangle \\ &= \alpha_1 |0 \dots 0\rangle + \dots + \alpha_{2^n} |1 \dots 1\rangle \\ &= \sum_{i=1}^{2^n} \alpha_i |e_i\rangle \end{aligned} \quad (2.12)$$

Where each α_i represents one amplitude of the basis states $|e_i\rangle$. For a n-qubit system 2^n possible combinations of $|0\rangle$ and $|1\rangle$ exist. These form the basis states of a 2^n -dimensional complex Hilbert Space. n-qubit states $|\psi\rangle$ are also normalized, so that $\sum_{i=1}^{2^n} |\alpha_i|^2 = 1$.

Example 3. Consider the following two single-qubit states:

$$\begin{aligned} |\psi_0\rangle &= \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \\ |\psi_1\rangle &= |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{aligned}$$

The state of a composite system with these two qubits is the tensor product of their

single-qubit states:

$$\begin{aligned}
 |\psi_c\rangle &= |\psi_0\rangle \otimes |\psi_1\rangle \\
 &= \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} \\
 &= \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|10\rangle
 \end{aligned}$$

For n-qubit systems the measurement rule from equation 2.3 extends to:

$$|\langle i\dots j|\psi\rangle|^2 = |\alpha_{i\dots j}|^2, \quad \text{with } i, \dots, j \in \{0, 1\} \quad (2.13)$$

Furthermore, it is possible to measure only a sub-system of a multi-qubit system, which is illustrated in the following example.

Example 4. Consider measuring only the first qubit of a two-qubit system in the following state:

$$|\psi\rangle = \frac{1}{\sqrt{4}}|00\rangle + \frac{1}{\sqrt{4}}|01\rangle + \frac{1}{\sqrt{4}}|10\rangle + \frac{1}{\sqrt{4}}|11\rangle$$

Assuming this measurement results in the first qubit to be in state $|0\rangle$, the post-measurement state of the entire system would be:

$$|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|01\rangle$$

Whereas the probabilities for the system being in one of the states with the first qubit in state $|1\rangle$ become zero, and the whole state is re-normalized.

2.5 ENTANGLEMENT

The multi-qubit states, shown so far, can all be obtained by the tensor product between the states of the qubits involved as described in equation 2.12. This implies that these states are separable. However, non-separable states exist, which cannot be represented by the tensor product. So-called entangled states are, for instance, Bell states which are

maximally entangled:

$$\begin{aligned}
 |\Phi^+\rangle &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\
 |\Phi^-\rangle &= \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \\
 |\Psi^+\rangle &= \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \\
 |\Psi^-\rangle &= \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)
 \end{aligned} \tag{2.14}$$

The unique feature of these states is that measuring one qubit also provides information about the state of the second qubit.

Example 5. Measuring the first qubit of the Bell state Φ^+ to be in state $|0\rangle$, would result in the post-measurement state:

$$|\phi^+\rangle_{post_measurement} = |00\rangle$$

Thus, after measurement, not only the first qubit's state would be known as $|0\rangle$, but also the state of the second one as $|0\rangle$.

Within entangled states, the measurement of a sub-system generally reveals information about the state of the non-measured part of the system.

2.6 PURE STATES, MIXED STATES, AND DENSITY OPERATOR

If the state of a quantum system is known precisely, such a state is called a pure state. States of systems with uncertainty about the system's state are called mixed states. Mixed states are described by density operators, otherwise known as density matrices. These represent probability distributions over different pure states in which a quantum system might find itself. Density matrices can also describe pure states. In this case, a density operator is obtained by the outer product of the state vector with itself.

$$\rho_{pure} = |\psi\rangle\langle\psi| \tag{2.15}$$

A mixed state's density operator is the weighted sum of pure states, in which the system might be.

$$\rho_{mixed} = \sum_i p_i |\psi_i\rangle\langle\psi_i| \tag{2.16}$$

Where p_i describes the probability of the system being in state $|\psi_i\rangle$. The following example illustrates how the density operators for pure and mixed states are obtained.

Example 6. A pure one-qubit state in superposition can be written in vector form as follows:

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

Computing the outer product of the vector with itself results in the following density matrix representation:

$$\rho_{pure} = |\psi\rangle\langle\psi| = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \begin{pmatrix} \alpha & \beta \end{pmatrix} = \begin{pmatrix} |\alpha|^2 & \alpha\beta^* \\ \alpha^*\beta & |\beta|^2 \end{pmatrix}$$

An example for a mixed state is the following probabilistic mixture of pure state density operators:

$$\rho_{mixed} = |\alpha|^2|0\rangle\langle 0| + |\beta|^2|1\rangle\langle 1| = \begin{pmatrix} |\alpha|^2 & 0 \\ 0 & |\beta|^2 \end{pmatrix}$$

Usually, all density operators are denoted by ρ without the subscript, regardless of whether the described state is pure or mixed.

Some important properties of density operators are:

- Hermiticity: $\rho^\dagger = \rho$
- Positive semi-definiteness: $\langle\phi|\rho|\phi\rangle \geq 0$
- Unit trace: $tr(\rho) = 1$

Quantum states cannot only be entirely pure or entirely mixed, but nuances exist between both. The less uncertainty about the state of a system there is the purer its state. The purity of a quantum state is defined as follows [16]:

$$purity(\rho) = tr(\rho^2) \tag{2.17}$$

The purity is a real number between zero and one. The smaller the purity is, the more mixed the state. If the purity equals one, the state is entirely pure.

Mixed states can be illustrated in the same fashion as pure states in the Bloch sphere, but with one difference. As shown in figure 2.1, pure states are located on the surface of the Bloch sphere. In contrast to that, mixed states are located inside the sphere. The more mixed the state is, the closer it is to the sphere's origin. Figure 2.2 shows an example of a

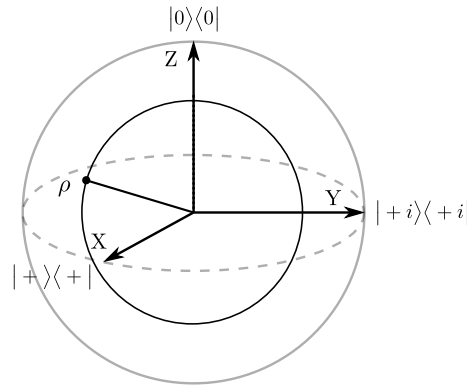


Figure 2.2: Illustration of a mixed state in the Bloch sphere. Mixed states are located inside the Bloch sphere and not on the surface. The closer a state is to the sphere's origin, the more mixed it is.

mixed state in the Bloch sphere.

Later in this thesis, it will be shown how density operators can be used to model word ambiguity and lexical entailment.

2.7 QUANTUM CIRCUITS

In order to run quantum programs on a quantum computer, the instructions to be executed are organized in so-called quantum circuits. Similar to classical computers, in quantum circuits, wires connect logical gates, which manipulate the state of qubits. The computational basis is the Z-basis, and at the beginning, all qubits are in state $|0\rangle$. Quantum circuits are visualized in circuit diagrams, with single wires symbolizing qubits, double lines denoting classical bits, and boxes representing gates and other unitary operations on one or more qubits.

Table 2.1 introduces the components of circuit diagrams with their diagrammatic and operator notation. Example 7 presents a complete circuit diagram. The first three operators in table 2.1 are the Pauli gates. These rotate the state of a qubit by π around the respective axes of the Bloch sphere. Since the computational basis is the Z-basis, the Pauli-X operator corresponds to a logical NOT gate. The Pauli gates are special cases of the more general parameterized rotation gates. These are the next three gates in the table. $RX(\theta)$, $RY(\theta)$ and $RZ(\theta)$ rotate the state around the respective axes by the angle θ . Figure 2.3 shows the effects of the RX, RY, and RZ gates on a state in the Bloch sphere.

The next gate in figure 2.1 is the so-called Hadamard gate. Starting from one of the

computational basis states, it generates an equal superposition:

$$|0\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{and} \quad |1\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$




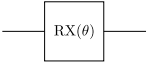
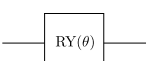


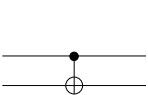

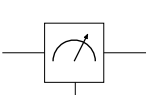
Name	Diagram	Operator
Pauli-X / NOT		$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Pauli-Y		$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
Pauli-Z		$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
RX(θ)		$\begin{pmatrix} \cos \frac{\theta}{2} & -i \sin \frac{\theta}{2} \\ -i \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{pmatrix}$
RY(θ)		$\begin{pmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{pmatrix}$
RZ(θ)		$\begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$
Hadamard		$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
CNOT		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
SWAP		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
Measurement		

Table 2.1: Overview of standard gates. The gate's name is on the left, the circuit diagram is located in the middle, and the corresponding operators can be found on the right.

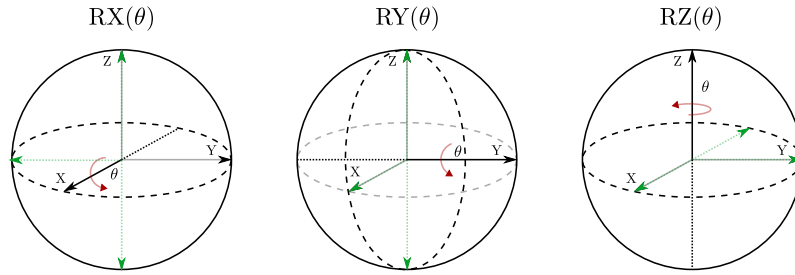
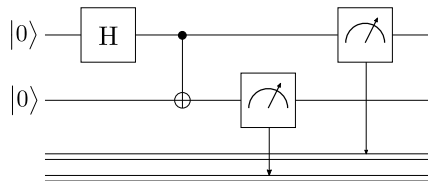


Figure 2.3: Illustration of the impact of the RX, RY, and RZ gates on a state in the Bloch sphere.

After the Hadamard gate, the table shows the CNOT gate, which operates on two qubits. The one with the smaller filled black dot is the control qubit. If the control qubit is in state $|1\rangle$ a NOT gate is performed on the other. This gate induces entanglement on two qubits. Example 7 illustrates this for one of the Bell states.

The next gate in the table is the SWAP gate, which swaps the states of two qubits. The last operation in figure 2.1 is the measurement, which is depicted by a box with a measurement device inside. A measurement is performed in the computational basis and the result is mapped onto a classical bit. The following example illustrates a full quantum circuit and its operations.

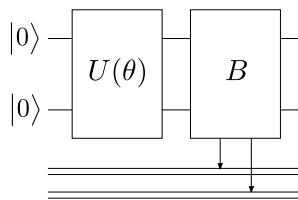
Example 7. The following circuit prepares the Bell state Φ^+ from equation 2.14:



The Hadamard gate transforms the first qubit's state from $|0\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. The state of the full system is then $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle$. Applying a CNOT gate to $|\psi\rangle$ results in the the maximally entangled Bell state $|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

After defining a quantum circuit, it can be compiled and executed on a simulator or a real quantum computer.

Since all quantum gates are unitaries, all gate operations in a quantum circuit can be summarized into the application of one unitary U to the initial state $|0..0\rangle$. This unitary is parameterized $U(\theta)$ if the circuit contains parameterized gates. The only non-unitary operation is the measurement. All measurements can be summarized into an observable B at the end of the circuit. This generalized circuit representation is depicted in a circuit diagram as follows:



2.8 VARIATIONAL QUANTUM CIRCUITS

Parameterized quantum circuits can be used as machine learning models for classification and regression tasks, similar to classical neural networks. These parameterized circuits are also called variational quantum circuits. The model's output is given by the expectation value of an observable B , and the parameter optimization is carried out on classical hardware.

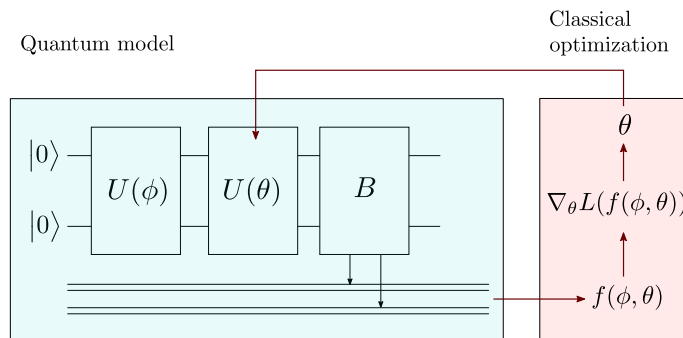


Figure 2.4: Scheme of a variational quantum circuit. The input data is encoded using the parameters ϕ , and the parameters θ are the trainable model parameters. The model's output is the expectation value $f(\phi, \theta)$ approximated by running the circuit many times and averaging the measurement results. The gradients are approximated numerically using the parameter shift rule in equation 2.19 and the parameters are updated using classical optimization techniques.

Figure 2.4 depicts the scheme for the training of such a circuit. The unitary describing the circuit's gate operations can be split into two parameterized unitaries. The first one is used to encode the input data. For example, the data can be encoded into the amplitudes of the quantum state. However, the details of data encoding routines will not be discussed here, as the quantum density matrix word embedding model proposed in this work does not require data encoding. For an introduction, please refer to the book [17].

The parameters θ of the second unitary are the trainable model parameters, which are updated using a classical optimizer. This part of a variational quantum circuit will be relevant later for learning quantum states corresponding to word meanings.

2.8.1 Gradients

In order to update the circuit parameters, the gradients of the quantum computations are needed. Choosing the expectation value of an observable B as output allows to approximate the gradients of the quantum computations on quantum hardware. However, individual measurement outcomes cannot be used because they are probabilistic. The expectation value of B , in contrast, is deterministic and can be estimated by averaging over many measurement results. Analytically the expectation value is computed as follows:

$$f(\theta) = \langle 0 | U^\dagger(\theta) B U(\theta) | 0 \rangle \quad (2.18)$$

Since it varies smoothly with small variations of the circuit parameters, it can be used to approximate the circuit's gradients numerically by the parameter-shift rule:

$$\nabla_{\theta_i} f = f(\theta_i + \epsilon) - f(\theta_i - \epsilon) \quad (2.19)$$

In practice, to approximate the gradient of a parameter, the circuits' expectation value is estimated with a slightly increased parameter value. Additionally, the same is done with a slightly decreased parameter value. Then the difference between the two is calculated to approximate the parameter's gradient. This is repeated for all circuit parameters.

On simulators, the gradients of equation 2.18 can be computed analytically.[18]

3

Distributional Compositional Model of Meaning

To computationally process natural language, it is necessary to find mathematical representations that consider both the meaning of words and the compositional structure of grammar. Most methods are only able to represent one of the two adequately. Distributional methods use vector spaces [1] to model word meanings, but do not take the grammatical structure of sentences into account. Thus, these cannot be used to represent sentence meanings as a whole. On the other hand, some models formalize grammar so that a sentence's grammatical validity is derived from the composition of the grammatical types of the component words [5]. But these models do not consider the word meanings and therefore cannot make any statement about the meaning of an entire sentence.

Clark et. al. [4] proposed a procedure to combine the two approaches by using the tensor product and connecting meaning vectors with their roles to find a unified model for sentence meanings. However, the sentence meanings in this model depend on the sentence's grammatical structure, which implies that the meanings of sentences with different grammatical structures do not live in the same vector space and cannot easily be compared to each other.

The distributional compositional model of meaning (DisCoCat model, [5]) also uses the tensor product and pairs meaning vectors with their grammatical types by combining the vector space model of meaning with Lambek's pregroup grammars [3], which share the same mathematical structure of compact closed categories. In contrast to [4] the DisCoCat model finds sentence meanings that live in the same vector space and are comparable by using standard distance measures, like the inner product.

Indeed, the mathematical structure of quantum mechanics within its category-theoretic formulation [8] inspired the development of the DisCoCat model. Therefore, quantum mechanical counterparts for the model's components can be found, which also brought up the idea of encoding word meanings into density matrices. Furthermore, this allows to carry out the model's computations on quantum computers, which has established the research field of quantum natural language processing.

This chapter introduces the distributional compositional model of meaning. First, it provides

the category-theoretic background. Then the distributional and compositional components are presented. Subsequently, the unification into one distributional compositional model for sentence meaning is shown. The end of the chapter demonstrates how the model's computations can be executed on quantum computers and what advantage can be expected from that.

3.1 CATEGORY-THEORETIC BACKGROUND

The DisCoCat model takes advantage of the fact that compact closed monoidal categories can capture both the quantitative meaning space of word vectors and the compositional structure of grammar to find one unified representation of sentence meaning. Therefore the underlying category theoretical concepts are introduced in this section. First, recall the basic definition of a category.

Definition 1. [19] A **category** C consists of a family of objects $ob(C)$, where

- for each pair of objects $A, B \in ob(C)$ there is a set of morphisms $C(A, B)$ from A to B
- for each triple $A, B, C \in ob(C)$ with morphisms $f \in C(A, B)$ and $g \in C(B, C)$, which can also be written as $f : A \rightarrow B$ and $g : B \rightarrow C$, there is a sequential composition:

$$g \circ f : A \rightarrow C$$

- the sequential composition is associative:

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

- for each object $A \in ob(C)$ there is an identity morphism $1_A : A \rightarrow A$, such that:

$$f \circ 1_A = f = 1_B \circ f$$

Definition 2. [5] A category is a (strict) **monoidal category** C if it has the following additional structure:

- a functor $\otimes : C \times C \rightarrow C$ called the tensor product, which is associative:

$$(A \otimes B) \otimes C = A \otimes (B \otimes C)$$

- a unit object I which satisfies: $I \otimes A = A = A \otimes I$

- a parallel composite $f \otimes g : A \otimes B \rightarrow C \otimes D$ for each ordered pair of morphisms $(f : A \rightarrow C, g : B \rightarrow D)$
- bifunctoriality:

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

The objects A of a monoidal category can be interpreted as types of systems. Elements of an object A of the category can also be considered morphisms of the form $\psi : I \rightarrow A$. The following example illustrates how monoidal categories will later be used to represent word meanings.

Example 8. The objects A will later be assigned Hilbert spaces for word meaning vectors of different grammatical roles. There will be a Hilbert space N for nouns and another Hilbert space S for sentences.

The meaning vectors for words or sentences of the respective types will then be elements of these Hilbert spaces of the form $\psi_N : I \rightarrow N$ and $\psi_S : I \rightarrow S$.

The word meaning vectors within a sentence will be composed using the vector tensor product as a monoidal functor. The sentence ‘John likes Mary’ will be composed as follows:

$$\vec{John} \otimes \vec{likes} \otimes \vec{Mary}$$

Where the verb ‘likes’ is an element of the composite Hilbert space $N^r \otimes S \otimes N^l$ indicating that it needs a noun on the left as a subject and another noun on the right as an object to produce a grammatically correct sentence. ‘John’ and ‘Mary’ are vectors in N . The whole sentence is then a tensor in the space $N \otimes N^r \otimes S \otimes N^l \otimes N$.

To map the sentence meaning from this composite space to the meaning space of sentences S , the specific morphisms of compact closed monoidal categories are used.

Definition 3. [5] A monoidal category is **compact closed** if for each object $A \in ob(C)$ there are also adjoint objects $A^r, A^l \in ob(C)$ with the following morphisms:

$$\eta^l : I \rightarrow A \otimes A^l \quad \epsilon^l : A^l \otimes A \rightarrow I \quad \eta^r : I \rightarrow A^r \otimes A \quad \epsilon^r : A \otimes A^r \rightarrow I \quad (3.1)$$

Which satisfy the following so-called snake equations.

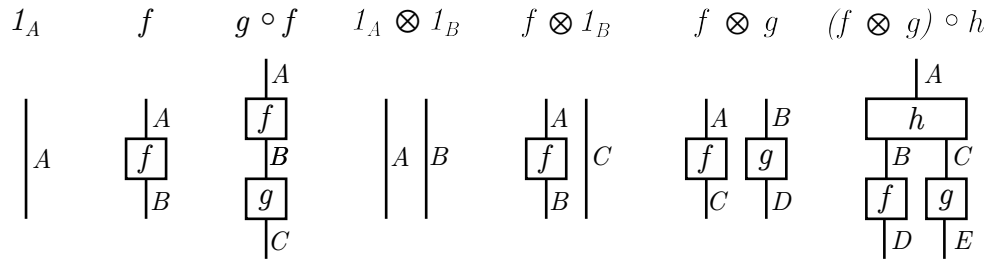
$$(1_A \otimes \epsilon^l) \circ (\eta^l \otimes 1_A) = 1_A \quad (\epsilon^r \otimes 1_A) \circ (1_A \otimes \eta^r) = 1_A \quad (3.2)$$

$$(\epsilon^l \otimes 1_{A^l}) \circ (1_{A^l} \otimes \eta^l) = 1_{A^l} \quad (1_{A^r} \otimes \epsilon^r) \circ (\eta^r \otimes 1_{A^r}) = 1_{A^r} \quad (3.3)$$

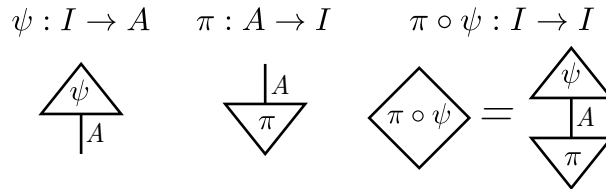
The ϵ morphisms map objects with their adjoint to the unit object allowing to map the tensor product of word meaning vectors above to the meaning space S of sentences: $N \otimes N^r \otimes S \otimes N^l \otimes N \rightarrow S$. This procedure will be explained in more detail in section 3.4.

3.1.1 Graphical Calculus

A particularly useful feature of monoidal categories is the graphical calculus that is designed, such that all valid equational statements between morphisms are also derivable in the graphical calculus. Moreover, it is not only applicable to monoidal categories but also to monoidal categories with some additional structure, like the abovementioned compact closed monoidal categories. Thus, it provides a diagrammatic calculus in which the word meanings can be combined with the grammatical structure. [5]



(a) The identity morphism is a straight line. General morphisms are boxes. Sequential composition is depicted by plugging the outgoing wire of the first morphism into the second morphism's incoming wire. Parallel composition is depicted as two lines or boxes next to each other.



(b) Morphisms without in- or output and their sequential composition

Figure 3.1: Graphical Calculus for monoidal Categories adapted from [5].

Graphical Calculus for Monoidal Categories

Figure 3.1 shows an overview of the graphical calculus. The diagrams are read from top to bottom. Subfigure 3.1 a depicts morphisms as boxes and the identity morphism as a straight line. Sequential (\circ) and parallel (\otimes) composition are built up from these components by arranging them one after the other and side by side. The unit object I is the empty diagram and therefore not depicted.

Subfigure 3.1 b shows the elements of an object A of a category, which are also morphisms $\psi : I \rightarrow A$ as triangles without input. Their duals are accordingly morphisms without output and depicted as flipped triangles. Composing a morphism without input and a morphism with no output results in a scalar, which in the diagrammatic calculus becomes a rhombus.

Translating these concepts into vector spaces, the morphisms in figure 3.1 b are vectors, their duals, and their inner product.

Compact Closed Categories

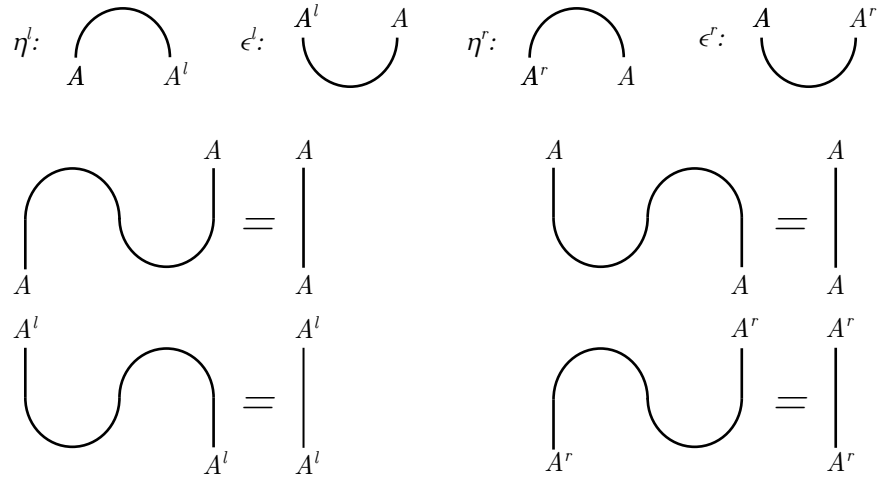
The morphisms of compact closed categories $\eta^r, \eta^l, \epsilon^r, \epsilon^l$ from equation 3.1 have their own additional graphical representation. They are depicted as wires in the form of caps and cups as shown in figure 3.2 a. This indicates that they connect types with their adjoints. Figure 3.2 a also shows the snake equations 3.2 and 3.3 in the same order. Their graphical depiction is explained using the first equation as an example:

$$(1_A \otimes \epsilon^l) \circ (\eta^l \otimes 1_A) = 1_A$$

This equation is depicted by the upper left diagram in figure 3.2 a. $(\eta^l \otimes 1_A)$ is shown in the upper part of the diagram by placing a cap and a single wire next to each other. The sequential composition of $(1_A \otimes \epsilon^l)$ is shown by adding a single wire and a cup underneath. As the equation states, this graphical notation can be simplified into a single straight wire corresponding to the identity map.

The reduction maps in the graphical calculus also bring the so-called swing rule with them, illustrated in figure 3.2 b. It states that an element of an object of the category composed with a cup can be swung around and thus equals the element's dual.

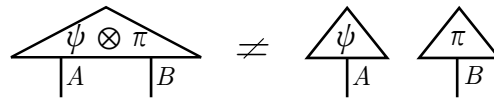
Figure 3.2 c shows another essential property of compact closed categories. Some morphisms composed in parallel are not separable. This will be relevant later because a transitive verb needs to interact with both its subject and its object, and thus its compound type cannot be decomposed into separate parts.



(a) Morphisms of compact closed categories from equation 3.1 and the snake equations 3.2 and 3.3 in the same order.



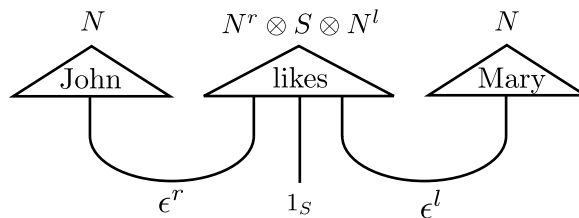
(b) Swing rule



(c) Some parallelly composed objects are not separable

Figure 3.2: Graphical Calculus for Compact Closed Categories, adapted from [5].

Example 9. The sentence ‘John likes Mary’ will be depicted in this graphical calculus as follows:



Where the triangles are the respective meaning vectors and the cups represent the grammatical structure to map the tensor product of word meaning vectors to the space S .

Here the tensor for ‘likes’ is an example of a non-separable morphism.

3.2 VECTOR SPACE MODEL OF MEANING

The word meanings within the DisCoCat model are described by vectors using the vector space model of meaning [1]. Chapter 4 will replace the vectors with density matrices, but the DisCoCat model is first introduced in its original form for a better understanding.

The meaning vectors are created based on the idea that ‘you shall know a word by the company it keeps’ by John Rupert Firth. A set of context words builds the basis for the vector space of word meanings. For example, these can be the 2000 most common words in the corpus. Each of these is assigned a basis vector. The vectors for the non-context words are obtained by counting the number of co-occurrences with the context words within a given window size. These constitute the word vectors’ entries at the positions corresponding to the respective context words. Mathematically, this is expressed in the following equation for the word vectors \vec{w} :

$$\vec{w} = \sum_{i=0}^N n_i \vec{e}_i \quad (3.4)$$

Where N is the number of context words, \vec{e}_i is the basis vector of context word i , and n_i is the number of co-occurrences of context word i with word w . The following example illustrates this procedure.

Example 10. When the context words ‘bark’, ‘sleep’ and ‘run’ build the following basis of a vector space:

$$\vec{bark} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \vec{sleep} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \vec{run} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

The following vectors represent the words ‘dog’ and ‘cat’:

$$\vec{dog} = \begin{pmatrix} 16 \\ 9 \\ 11 \end{pmatrix}, \quad \vec{cat} = \begin{pmatrix} 3 \\ 12 \\ 10 \end{pmatrix}$$

These vectors indicate that ‘dog’ occurs more often in the context of bark, while cats are more strongly associated with ‘sleep’, and both are similarly much associated with ‘run’. Their similarity is calculated by the inner product or the cosine similarity.

This method finds word meaning representations based on their distribution throughout the text. Approaches like this are also referred to as distributional methods in natural language processing.

One drawback is that the counting of co-occurrences cannot be directly extended to calculate the meaning of longer phrases or sentences because these usually do not occur repeatedly.

3.3 LAMBEK'S PREGROUP GRAMMAR

The other component of the DisCoCat model that is responsible for the grammatical structure is Lambek's pregroup grammar. In [3] Lambek introduced the idea to model the compositional structure of grammar using pregroups. The definition of pregroups is based on partially ordered monoids.

Definition 4. [3] A **partially ordered monoid** $(P, \leq, \cdot, 1)$ is a set P with

- a binary relation \leq , which is
 - reflexive: $a \leq a$
 - antisymmetric: if $a \leq b$ and $b \leq a$, then $a = b$
 - transitive: if $a \leq b$ and $b \leq c$, then $a \leq c$
- a binary operation \cdot , which is
 - associative: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
 - has an identity element: $1 \cdot a = a \cdot 1 = a$

A pregroup is then defined as follows:

Definition 5. [5] A **pregroup** $(P, \leq, \cdot, 1, (-)^l, (-)^r)$ is a partially ordered monoid, where every element $p \in P$ has a left adjoint p^l and a right adjoint p^r , such that:

$$\begin{aligned} p^l \cdot p &\leq 1 \leq p \cdot p^l \\ p \cdot p^r &\leq 1 \leq p^r \cdot p \end{aligned}$$

The elements $p \in P$ are called pregroup types. In the case of grammar, these are some fixed basic grammatical roles, like nouns or sentences. Additional compound types are formed from these basic roles by combining them with adjoints. This allows words like transitive verbs to demand certain grammatical types on their right or left. In order to inspect a sentence's grammatical validity, each word is assigned its corresponding type (basic or compound), and subsequently, pregroup reductions \leq are applied. Grammatically

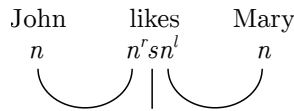
correct sentences reduce to the type s . The following example illustrates this procedure. From now on, the notation \leq for reductions is replaced by arrows \rightarrow , and the dot \cdot between the types is neglected.

Example 11. [5] **Transitive sentences**

English transitive sentences are modeled in the framework of pregroups by fixing the following basic types:

- n : noun
- j : infinitive verb
- s : declarative statement
- σ : glueing type

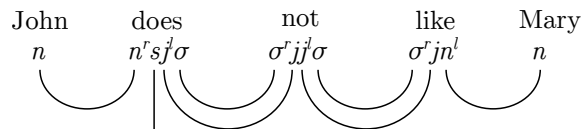
Each word in the sentence, e.g., ‘John likes Mary’, is assigned the corresponding type. The subject and the object are nouns with the basic type n . ‘Like’ is a transitive verb, which needs a subject on the left and an object on the right to produce a grammatically correct sentence. Thus it is assigned the compound type $n^r s n^l$. The adjoints of type n interact with the subject and the object so that only the type s remains. For the sentence ‘John likes Mary’, this is depicted as follows:



The sentence’s pregroup reductions are calculated as follows:

$$n(n^r s n^l)n \rightarrow 1s n^l n \rightarrow 1s1 \rightarrow s$$

Since the sentence’s grammatical types reduce to s , the sentence is grammatically correct. For a negative transitive sentence, the following types are assigned to its components:



From calculating the sentence’s reductions, it follows that this sentence is grammatically correct as well:

$$n(n^r s j^l \sigma)(\sigma^r j j^l \sigma)(\sigma^r j n^l)n \rightarrow s$$

3.4 CATEGORY-THEORETIC COMBINATION OF GRAMMAR AND MEANING

Lambek’s pregroup grammar formalizes a sentence’s grammatical structure. The vector space model of meaning provides information about the meaning of words within a text corpus. Thus, the DisCoCat model combines the two to find a sentence’s meaning based on the meaning of its component words and its grammatical structure. This unification is possible because both finite-dimensional Hilbert spaces and pregroups share the mathematical structure of compact closed categories.

3.4.1 Vector Spaces as Compact Closed Categories

The vector spaces spanned by the context words in the vector space model of meaning can be considered as objects of the category **FHilb** of finite-dimensional real Hilbert spaces, and the meaning vectors themselves are elements of these Hilbert spaces. The morphisms in **FHilb** are linear maps, and the monoidal functor is the vector tensor product. For a Hilbert space V with basis $\{|e_i\rangle\}_i$ the caps and cups have the following form:

$$\begin{aligned} \eta^l = \eta^r : \mathbb{R} &\rightarrow V \otimes V \\ 1 &\mapsto \sum_i |e_i\rangle \otimes |e_i\rangle \end{aligned} \quad (3.5)$$

$$\begin{aligned} \epsilon^l = \epsilon^r : V \otimes V &\rightarrow \mathbb{R} \\ \sum_{ij} c_{ij} |v_i\rangle \otimes |w_j\rangle &\mapsto \sum_{ij} c_{ij} \langle v_i | w_j \rangle \end{aligned} \quad (3.6)$$

These morphisms fulfill the snake equations 3.2 and 3.3. Since the tensor product is commutative $V \otimes W \cong W \otimes V$, the left and right morphisms are equal, and the left and right adjoints of an object V in **FHilb** are both equal to the object itself: $V^l = V^r = V^* = V$. [5] Thus, V^* is used to denote both. In the context of language, this means that the order of words is ignored, and the category of finite-dimensional Hilbert spaces alone cannot reflect a sentence’s grammatical structure. [20] Non-commutative caps and cups are needed from the category of pregroups to unite the meaning vectors with grammatical structure.

3.4.2 Pregroups as Compact Closed Categories

The structure of compact closed categories is given within the definition of pregroups. A Pregroup **P** is a partially ordered monoid. The partial order provides the structure to

consider a pregroup as a category. The objects of this category are the basic grammatical types. The morphisms map grammatical types to other grammatical types ($A \rightarrow B$). The monoidal structure is determined because it is a monoid with a monoid multiplication as a tensor. This allows composing the basic grammatical types into compound types, as seen before for transitive verbs $n^r sn^l$. The pregroup reductions with the left and right adjoints provide the maps from which compact closure arises:

$$\begin{aligned} \eta^l &= [1 \leq p \cdot p^l] & \epsilon^l &= [p^l \cdot p \leq 1] \\ \eta^r &= [1 \leq p^r \cdot p] & \epsilon^r &= [p \cdot p^r \leq 1] \end{aligned} \tag{3.7}$$

These cups and caps also fulfill the snake equations 3.2 and 3.3 [5]. In contrast to the category **FHilb**, the monoidal tensor of the category **P** is not commutative, and the left and right adjoints and cups and caps are not the same. Thus this category is able to reflect the order of words within a sentence and its grammatical structure.

3.4.3 Sentence Meanings

These two categories, of which one contains the structure for word meaning vectors and the other contains the grammatical structure of a sentence, can be combined to benefit from the features of both. To do so, a strong monoidal functor Q , which preserves the compact closed structure, is used to map the pregroup reductions from **P** onto the category **FHilb** [20]:

$$Q : \mathbf{P} \rightarrow \mathbf{FHilb}$$

Pregroup types like n and s for nouns and sentences are mapped to finite-dimensional Hilbert spaces.

$$Q(n) = N \qquad Q(s) = S$$

Composite types, like that of transitive verbs $n^r sn^l$ are constructed using the monoidal functor in **FHilb**.

$$Q(n^r sn^l) = N \otimes S \otimes N$$

This is an example where morphisms composed in parallel are not separable (see section 3.1.1 and figure 3.2). Splitting the compound type of a transitive verb into its components does not make sense since it needs to interact with both a subject and an object. For example, in the sentence ‘John drinks juice.’, the verb interacts with ‘John’ and ‘juice’ to

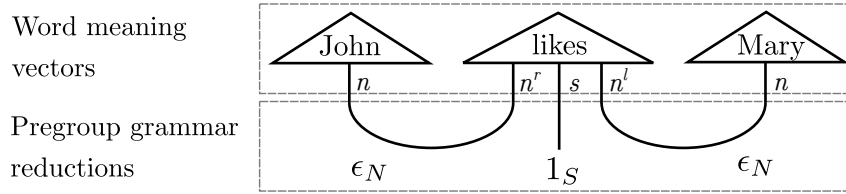
build a sentence.

The morphisms α of pregroups are mapped to linear maps that interpret sentences with the corresponding grammatical structure. The meaning of a sentence $w_1 w_2 \dots w_n$, where the word meanings are given as vectors $|w_i\rangle$ and their grammatical types are given as pregroup types p_i together with their reduction $\alpha : p_1, p_2, \dots, p_n \rightarrow s$, can be expressed as follows:

$$|w_1 w_2 \dots w_n\rangle = Q(\alpha)(|w_1\rangle \otimes |w_2\rangle \otimes \dots \otimes |w_n\rangle) \quad (3.8)$$

The following example illustrates what this means in practice.

Example 12. [20] The sentence ‘John likes Mary’ is depicted as follows in the DiCoCat model:



The nouns ‘John’ and ‘Mary’ are vectors in the Hilbert space N . The verb ‘likes’ is a rank three tensor in the Hilbert space $N \otimes S \otimes N$ and the entire sentence ‘John likes Mary’ is a tensor in the Hilbert space $N \otimes (N \otimes S \otimes N) \otimes N$. The pregroup reduction corresponding to the grammatical structure is the following linear map:

$$\epsilon_N \otimes 1_S \otimes \epsilon_N : N \otimes (N \otimes S \otimes N) \otimes N \rightarrow S$$

The tensor for the transitive verb $|likes\rangle \in N \otimes S \otimes N$ can be written as sum of the subjects n_i and objects n_k it connects in the sentences s_j :

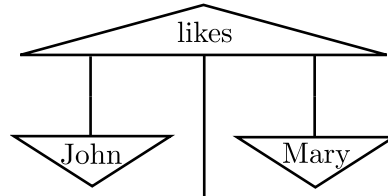
$$|likes\rangle = \sum_{ijk} c_{ijk} |n_i\rangle \otimes |s_j\rangle \otimes |n_k\rangle$$

The meaning of the sentence is then calculated as follows:

$$\begin{aligned} |John\ likes\ Mary\rangle &= \epsilon_N \otimes 1_S \otimes \epsilon_N (|John\rangle \otimes |likes\rangle \otimes |Mary\rangle) \\ &= \sum_{ijk} c_{ijk} \langle John | n_i \rangle \otimes |s_j\rangle \otimes \langle n_k | Mary \rangle \\ &= \sum_j \sum_{ik} c_{ijk} \langle John | n_i \rangle \langle n_k | Mary \rangle |s_j\rangle \end{aligned}$$

Alternatively, the sentence diagram can be simplified initially using the graphical calculus.

Applying the swing rule from figure 3.2 c yields the following form for ‘John likes Mary’, where the triangles of ‘John’ and ‘Mary’ are flipped around. [5]:



The flipped triangles are vectors in the dual space and are written as Bra’s using the Dirac notation. This diagram corresponds to the following equational meaning of the sentence:

$$(\langle John| \otimes 1_S \otimes \langle Mary|) \circ |likes\rangle$$

The vector of the verb ‘likes’ is sequentially composed with the tensor product of the dual of ‘John’, the identity in the Hilbert space S , and the dual of Mary. Evaluating this expression for concrete vectors is much easier than the expression above.

Here one great advantage of the graphical calculus can be seen. Using the diagrammatic transformations often provides simpler expressions to compute.

3.5 GOING QUANTUM

The computations within the DisCoCat model take place in high-dimensional tensor product spaces and quickly exceed the classically available computational resources. Here quantum computers prove useful. These are naturally suited to perform computations in high-dimensional tensor product spaces.

Indeed this is not a coincidence: the DisCoCat model was originally inspired by quantum protocols like quantum teleportation [7] in the formalism of categorical quantum mechanics [8]. A complete introduction to categorical quantum mechanics is provided in [21]. The entire theory of quantum mechanics can also be formulated within the framework of compact closed monoidal categories.

Since they share the same mathematical structure, quantum mechanical counterparts can be found for the elements of the DisCoCat model. Comparing the form of word meaning vectors from equation 3.4, $\vec{w} = \sum_{i=0}^N n_i \vec{e}_i$, to the definition of pure quantum states in superposition from equation 2.12, $|\psi\rangle = \sum_{i=1}^{2^n} \alpha_i |e_i\rangle$, shows that the two are structurally similar. Word meaning vectors can therefore be easily transferred into pure quantum states. The co-occurrence frequencies n_i are then encoded into the state’s amplitudes α_i , and the

basis vectors for the context words \vec{e}_i , are encoded into the basis states of the quantum system $|e_i\rangle$. Thus, word meaning vectors are then quantum states and their duals are the duals of quantum states, also called quantum effects or measurements. The caps from the pregroup reductions can be interpreted as Bell states from equation 2.14 and cups as Bell measurements. The following example illustrates the circuit for the sentence ‘John likes Mary’.

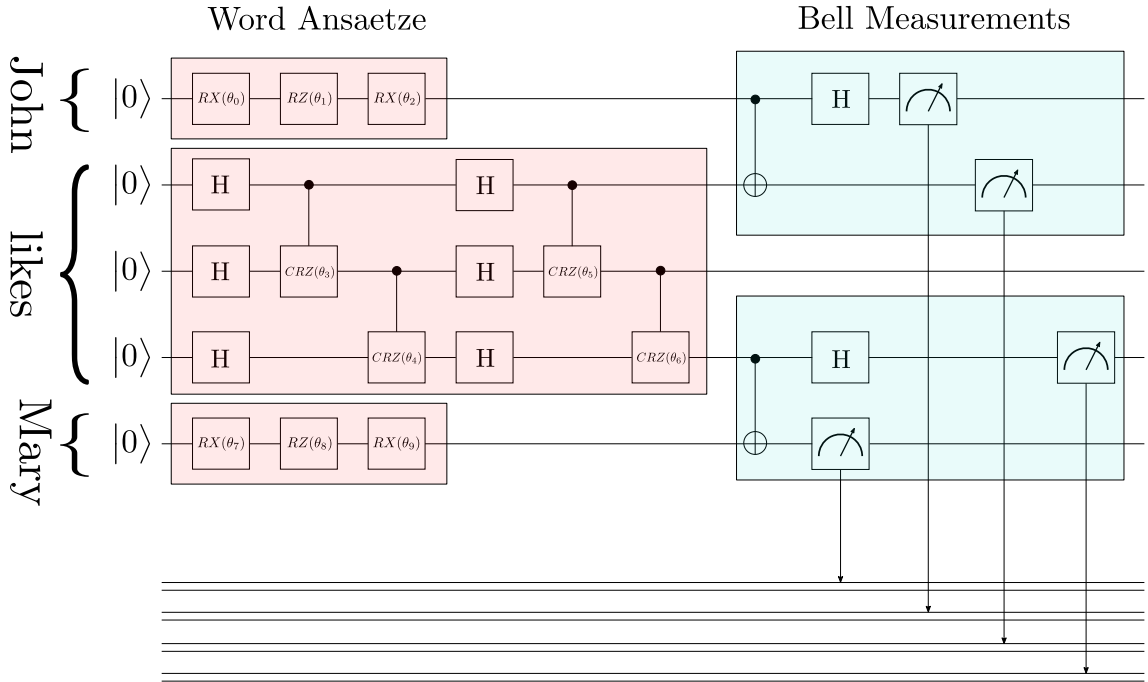


Figure 3.3: Quantum circuit for the sentence ‘John likes Mary’. A sequence of parameterized rotation gates prepares the single-qubit states of the nouns and the three-qubit state of the transitive verb is prepared by two IQP-layers consisting of a row of Hadamard gates followed by a chain of controlled Z rotations each. At the end of the circuit, the Bell measurements corresponding to the cups are applied.

Example 13. Figure 3.3 shows a possible quantum circuit diagram for the sentence ‘John likes Mary’. Each word is represented by an n -qubit state in the Hilbert space corresponding to its grammatical type. For each basic type, a number of qubits is chosen to define its Hilbert space’s dimension. In this example, nouns and sentences are assigned one qubit, respectively. The transitive verb is assigned three qubits since it has the compound type $n^r sn^l$ in the Hilbert space $N \otimes S \otimes N$.

Furthermore, circuit ansaetze have to be chosen for the word types. In this example, a sequence of three parameterized rotation gates (RX, RZ, RX) is chosen for the single-qubit

	One transitive verb	10k transitive verbs
Classical	8×10^9 bits	8×10^{13} bits
Quantum	33 qubits	47 qubits

Table 3.1: This table presents the reduction of necessary qubits when using quantum hardware compared to bits on classical hardware.

states, and the so-called IQP-ansatz is selected for the verb’s three-qubit state. An IQP-layer consists of a row of Hadamard gates on all qubits followed by a chain of controlled X rotations. In this example, two IQP-layers are used. At the end of the circuit, the Bell measurements that correspond to the cups are applied. These are constructed similarly to a Bell state (see example 7), but the Hadamard gate comes after the CNOT gate.

These choices are not fixed, but the basic types can be assigned any other qubit number, and different ansaetze can be chosen for the word’s state preparations.

Such a circuit can either be used to compose already known word states together for calculating the sentence’s meaning or as a variational circuit. In this case, a training dataset of sentences and their respective meanings as labels are needed, and the circuit parameters are trained to fit the expected sentence meaning. Afterwards, the meaning of sentences built from the same vocabulary can be calculated from the learned word meanings.

3.5.1 Quantum Advantage

The primary quantum advantage of implementing the DisCoCat model on currently available quantum computers arises from the efficient encoding of high-dimensional vectors. Since quantum states are composed by the tensor product, n-qubit systems have 2^n degrees of freedom, and thus N-dimensional vectors can be stored in $\log_2 N$ qubits. The following example illustrates the saved computational resources for transitive verbs.

Example 14. [6] Transitive verbs live in the vector space $N \otimes S \otimes N$. Assuming the basis of the meaning space N consists of the 2000 most common nouns in the text corpus and the meaning spaces S and N have the same dimension, this results in a $2000^3 = 8 \times 10^9$ dimensional meaning space for transitive verbs. Table 3.1 shows the enormous reduction of necessary qubits for storing transitive verbs compared to classical bits.

An additional advantage could be gained from importing the word meanings into a quantum RAM. The word meaning states could be retrieved with complexity linear in the number of qubits [22]. Unfortunately, this QRAM is still a topic of ongoing research and thus is more a hypothetical future advantage. For now, the quantum advantage entirely arises from the tensor product composition of quantum systems.[6]

4

Word Meaning with Density Matrices

The previous chapter showed how word meaning vectors could be encoded into pure quantum states. However, word meanings can alternatively be expressed as mixed quantum states (see section 2.6) [20]. Recall that these are described by density matrices and represent a statistical mixture of pure states when there is uncertainty about the system's state. In contrast, superposition is an inherent property of pure quantum states that does not describe a lack of knowledge about the system but the fact that the system can be in intermediate states between the basis states.

Using mixed states adds another level that allows representing the meaning of words that subsume multiple concepts, like the ambiguous word 'bank'. The basis states represent contextual features. Different pure states in superposition describe different meanings of the word in different contexts, and the mixed state's density matrix reflects a probability distribution over these different underlying meanings. A word is then described by the following equation:

$$\rho_w = \sum_i p_i |w_i\rangle\langle w_i| \quad \text{with} \quad |w_i\rangle = \sum_j \alpha_j |c_j\rangle \quad (4.1)$$

Where $|w_i\rangle\langle w_i|$ are the density matrices of the pure states $|w_i\rangle$. These are in superposition of the contextual features $|c_j\rangle$ and α_j describe the pure states' amplitudes. p_i describes the probability that the word w means the underlying meaning w_i .

This proves helpful in describing word ambiguity and lexical entailment, where words subsume multiple concepts. Before knowing a word's context, the word can only be thought of as a probability distribution over possible meanings. The following examples illustrate the use of density matrices to represent these phenomena.

Example 15. Word Ambiguity

Some words are ambiguous. When reading the word 'bank' without context, it cannot be known whether the 'financial institute', the 'sandbank', or the 'river bank' is meant. Therefore 'bank' is best modeled as a probability distribution over its possible meanings,

where p_i are the respective probabilities.

$$\rho_{bank} = p_0 |river\ bank\rangle\langle river\ bank| + p_1 |financial\ institute\rangle\langle financial\ institute| + p_2 |sandbank\rangle\langle sandbank|$$

The meaning of the words $|river\ bank\rangle$, $|financial\ institute\rangle$ and $|sandbank\rangle$ themselves are pure quantum states in superposition over their contextual features c_i , e.g. the state $|river\ bank\rangle$ could be described by the following state:

$$|river\ bank\rangle = \frac{1}{2}|river\rangle + \frac{1}{4}|bikeway\rangle + \frac{1}{4}|promenade\rangle$$

This state is then multiplied with its own dual to receive its density matrix representation:

$$\rho_{river\ bank} = |river\ bank\rangle\langle river\ bank|$$

The states for $|financial\ institute\rangle$ and $|sandbank\rangle$ are obtained analogously.

Example 16. [20] Lexical Entailment

Words can also be in a hierarchical relationship. For example, the word ‘pet’ is the umbrella term for all animals that people keep at home, like dogs, cats, hamsters, birds, etc. When reading the word ‘pet’ without any context, it cannot be known which animal is meant. Thus hyperonyms like ‘pet’ can also be modeled by a probability distribution over the meanings of the sub-terms (hyponyms).

$$\rho_{pet} = p_0 |dog\rangle\langle dog| + p_1 |cat\rangle\langle cat| + p_2 |hamster\rangle\langle hamster| + p_3 |bird\rangle\langle bird|$$

Like in the previous example, the contributing terms’ meaning, like $|dog\rangle$, are pure quantum states in superposition, which are then multiplied with their duals to obtain their density matrix representation.

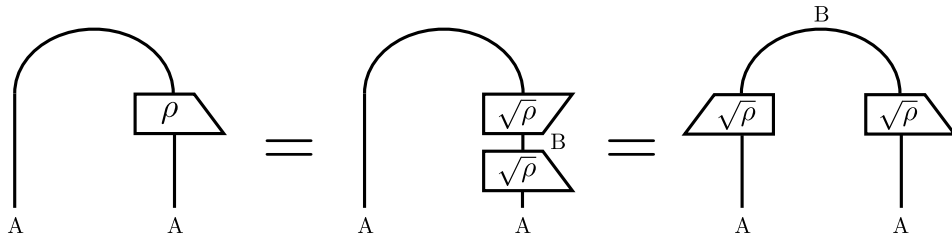
This chapter first integrates density matrices into the DisCoCat model by considering them from a category-theoretic point of view. Subsequently, similarity measures for density matrices are introduced, and their ability to capture lexical entailment and word ambiguity is shown. Additionally, two alternative composition methods for density matrices are shortly presented. Finally, the classical neural network architecture Word2DM for learning density matrix word embeddings, an adaption of the Word2Vec model, is explained.

4.1 DENSITY MATRICES AS COMPACT CLOSED CATEGORY

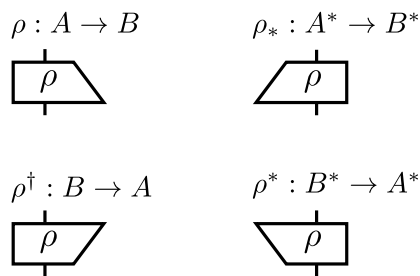
In order to use density matrices in the DisCoCat model, a compact closed category with density matrices as elements and morphisms between density matrices is needed. Since density matrices are also elements of finite-dimensional Hilbert spaces, these form the objects of the new category. Recall that elements of an object A of a category can also be written as morphisms of the form $\psi : I \rightarrow A$ (see figure 3.1 b). Thus, density matrices can be written as $\lceil \rho \rceil : I \rightarrow A^* \otimes A$. In order to find a candidate for the morphisms of the new category, it is useful to take a closer look at the properties of density matrices in operator form. A density operator of a pure state $|w\rangle\langle w|$ has the form $\phi \circ \phi^\dagger : A \rightarrow A$ with $\phi^\dagger \circ \phi = id_I$. These are thus positive operators by definition.

Definition 6. [12] **Positive Operator**

An operator $\rho : A \rightarrow A$ is positive if and only if there exists another operator $\sqrt{\rho}$ such that $\rho = \sqrt{\rho}^\dagger \circ \sqrt{\rho}$. The cap η is the isomorphism between the two representations $\rho : A \rightarrow A$ and $\lceil \rho \rceil : I \rightarrow A^* \otimes A$. Thus, the graphical depiction of positive operators is:



Here, the boxes of morphisms are replaced by trapeziums to make the conjugates and adjoints of the morphisms visible.



General density operators, $\rho = \sum_i p_i |w_i\rangle\langle w_i|$ are also positive operators of the form $\rho : A \rightarrow A$ [23]. Therefore, the morphisms between density matrices $A^* \otimes A \rightarrow B^* \otimes B$ have to preserve positivity. Completely positive maps are well-suited for this task.

Definition 7. [12] **Completely positive maps**

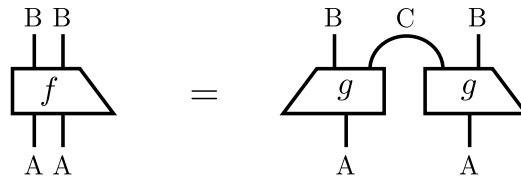
A morphism f is a positive map if it preserves positivity for any positive operator $\rho : \rho \geq 0$

$0 \Rightarrow f(\rho) \geq 0$. It is said to be completely positive if $(id_V \otimes f)A$ is positive for any positive operator A and any space V .

The graphical depiction of completely positive maps is given by the Stinespring Dilation Theorem.

Theorem 1. [12] Stinespring Dilation Theorem

A morphism $f : A^* \otimes A \rightarrow B^* \otimes B$ is completely positive if and only if an object C and a morphism $g : C \otimes B$ exist such that the following graphical equation holds:



At this point, all components for a new category **CPM(FHilb)** have been introduced. The objects are once more finite-dimensional Hilbert spaces, but their elements are now density matrices $\lceil \rho \rceil : I^* \otimes I \rightarrow A^* \otimes A$. The morphisms in this new category are completely positive maps $A^* \otimes A \rightarrow B^* \otimes B$. [12]

CPM(FHilb) is a monoidal category with a completely positive identity morphism $1_{A^* \otimes A} : A^* \otimes A \rightarrow A^* \otimes A$. The sequential composite $g \circ f$ of completely positive maps $f : A^* \otimes A \rightarrow B^* \otimes B$ and $g : B^* \otimes B \rightarrow C^* \otimes C$ is also completely positive. The monoidal tensor is the tensor product of density matrices $f \otimes g$ and is completely positive as well. [23]

CPM(FHilb) is compact closed. As in **FHilb**, the adjoints are equal to the objects of the category: $A^r = A^l = A^* = A$. The maps from which compact closure arises are defined as follows:

$$\begin{aligned}
 \eta^l &= (\eta_{FHilb}^r \otimes \eta_{FHilb}^l) \circ (1_A \otimes \sigma \otimes 1_A) \\
 \eta^r &= (\eta_{FHilb}^l \otimes \eta_{FHilb}^r) \circ (1_A \otimes \sigma \otimes 1_A) \\
 \epsilon^l &= (1_A \otimes \sigma \otimes 1_A) \circ (\epsilon_{FHilb}^r \otimes \epsilon_{FHilb}^l) \\
 \epsilon^r &= (1_A \otimes \sigma \otimes 1_A) \circ (\epsilon_{FHilb}^l \otimes \epsilon_{FHilb}^r)
 \end{aligned} \tag{4.2}$$

where σ is the swap map $\sigma(v \otimes w) = (w \otimes v)$.

The concrete maps have the following form:

$$\begin{aligned}
 \eta^l = \eta^r &: \mathbb{R} \rightarrow (A \otimes A) \otimes (A \otimes A) :: 1 \mapsto \sum_i |e_i\rangle \otimes |e_i\rangle \otimes \sum_j |e_j\rangle \otimes |e_j\rangle \\
 \epsilon^l = \epsilon^r &: (A \otimes A) \otimes (A \otimes A) \rightarrow \mathbb{R} :: \sum_{ijkl} c_{ijkl} |v_i\rangle \otimes |w_j\rangle \otimes |u_k\rangle \otimes |p_l\rangle \mapsto \sum_{ijkl} c_{ijkl} \langle v_i | u_k \rangle \langle w_j | p_l \rangle
 \end{aligned}$$

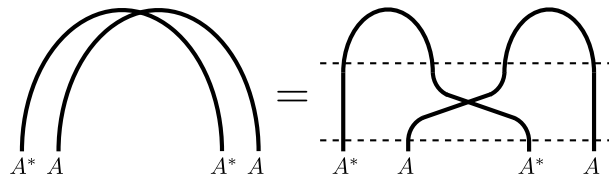
(4.3)

In the graphical notation of **FHilb**, the components of **CPM(FHilb)** are depicted by doubling the objects and wires as shown in the right column in table 4.1. The notation is simplified by replacing the doubled wires with single thick wires, as shown in the left column. Objects $A^* \otimes A$ are then also simply written as A in **CPM(FHilb)** and morphisms $f : A^* \otimes A \rightarrow B^* \otimes B$ as $f : A \rightarrow B$.

	CPM(FHilb)	FHilb
identity 1		
caps η	⤿	⤿⤿
cups ϵ	⤵	⤵⤵
pure density matrices	⤴	⤴ ⤴
mixed density matrices	⤴	⤴⤴

Table 4.1: Overview of the graphical notations of the components of **CPM(FHilb)** with their pendant in **FHilb**. Doubled wires in **FHilb** are simplified to single thick wires in **CPM(FHilb)**.

This depiction can be justified by comparing the form of the morphisms in **CPM(FHilb)** with their form in **FHilb**. The identity map in **CPM(FHilb)** $1_A : A \rightarrow A$ is a single thick wire corresponding to two wires representing its form $1_A : A^* \otimes A \rightarrow A^* \otimes A$ in **FHilb**. The representation of the cups can be clarified by bending the wires as follows:



This picture corresponds to sequentially composing the tensor product of two caps with the tensor product of an identity, a swap map, and another identity, which is exactly how

the η morphisms are defined in equation 4.2.

Density matrices $I \rightarrow A$ in $\mathbf{CPM}(\mathbf{FHilb})$ correspond to morphisms $I^* \otimes I \rightarrow A^* \otimes A$ in \mathbf{FHilb} . Thus, these are depicted by doubling the triangles, where the trapeziums again indicate the adjoints. Pure states correspond to separable states in \mathbf{FHilb} and mixed states correspond to non-separable states in \mathbf{FHilb} . The non-separable states have the depicted internal wiring. With these thick wires, all rules of the original graphical calculus are applicable. [12]

4.2 MEANING OF SENTENCES IN $\mathbf{CPM}(\mathbf{FHilb})$

The previous section established that the category $\mathbf{CPM}(\mathbf{FHilb})$ is also compact closed and shares the same diagrammatic calculus as \mathbf{FHilb} , but with thick wires. Therefore, the word meanings in $\mathbf{CPM}(\mathbf{FHilb})$ can be combined with the grammatical structure from pregroups for calculating a sentence's meaning in the DisCoCat model. This is done following section 3.4.3 by using a strong monoidal functor $S : P \rightarrow \mathbf{CPM}(\mathbf{FHilb})$. The pregroup types are now mapped to spaces in $\mathbf{CPM}(\mathbf{FHilb})$, i.e. the mapping of the noun type n is $S(n) = N$, where $N \in \mathbf{CPM}(\mathbf{FHilb})$. The morphisms α of pregroups are mapped to completely positive maps in $\mathbf{CPM}(\mathbf{FHilb})$. The meaning of a sentence w_1, w_2, \dots, w_n , where the word meanings are given as density matrices $\widehat{w}_i : I \rightarrow S(p_i)$, and their grammatical types are given as pregroup types p_i together with their reduction $\alpha : p_1, p_2, \dots, p_n \rightarrow s$, can be written as:

$$w_1 \widehat{w_2 \dots w_n} = S(r)(\widehat{w}_q \otimes \dots \otimes \widehat{w}_n) \quad (4.4)$$

The hats indicate that the meaning of the words w_i are now represented by density matrices [20]. The following example illustrates this procedure in the category $\mathbf{CPM}(\mathbf{FHilb})$.

Example 17. [20] The noun space N is a real Hilbert space with the basis vectors $\{|n_i\rangle\}_i$. The nouns below are then vectors in that space, which can be a linear combination of the basis states. Or, using quantum terminology, they are pure states in a superposition of the basis states.

Suppose a text corpus contains three sisters Anny, Betty and Clara. They all enjoy hot drinks. Occasionally, they like drinking a cup of tea. But what they appreciate most is a good cup of coffee. 'Anny', 'Betty', 'Clara', 'tea' and 'coffee' are pure states in Hilbert space N :

$$|Annie\rangle = \sum_i a_i |n_i\rangle, |Betty\rangle = \sum_i b_i |n_i\rangle, |Clara\rangle = \sum_i c_i |n_i\rangle$$

$$|tea\rangle = \sum_i d_i |n_i\rangle, |coffee\rangle = \sum_i e_i |n_i\rangle$$

The transitive verbs ‘like’ and ‘appreciate’ are given by the following tensors:

$$|like\rangle = \sum_{pqr} C_{pqr} |n_p\rangle \otimes |s_q\rangle \otimes |n_r\rangle$$

$$|appreciate\rangle = \sum_{pqr} D_{pqr} |n_p\rangle \otimes |s_q\rangle \otimes |n_r\rangle$$

The following words are then a probabilistic mixture of the word vectors introduced so far. Or, using quantum terminology again, they are mixed states represented by the following density matrices:

$$\widehat{The\ sisters} = \frac{1}{3}(|Annie\rangle\langle Annie| + |Betty\rangle\langle Betty| + |Clara\rangle\langle Clara|)$$

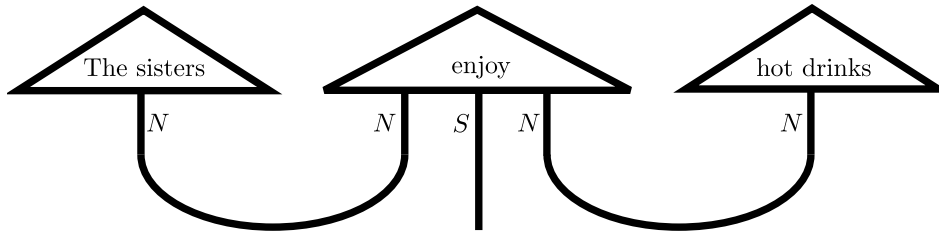
$$\widehat{hot\ drinks} = \frac{1}{2}(|tea\rangle\langle tea| + |coffee\rangle\langle coffee|)$$

$$\widehat{enjoy} = \frac{1}{2}(|like\rangle\langle like| + |appreciate\rangle\langle appreciate|)$$

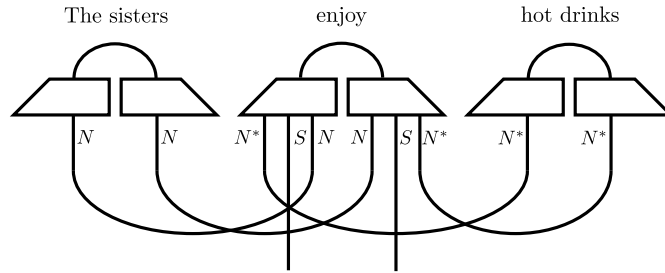
For simplicity, the adjective-noun phrase ‘hot drinks’ is considered as one word of type n in this example. The meaning of the sentence ‘The sisters enjoy hot drinks’ can then be calculated as follows:

$$The\ sisters\ \widehat{enjoy}\ hot\ drinks = (\epsilon_N \otimes 1_S \otimes \epsilon_N)(\widehat{The\ sisters} \otimes \widehat{enjoy} \otimes \widehat{hot\ drinks})$$

Its graphical depiction in **CPM(FHilb)** looks exactly the same as before, but with thick wires now:



Using the doubled notation in **FHilb** this corresponds to the following graphical depiction:



The mixed states are depicted by doubled morphisms connected by a cap, and the doubled cups are sequentially composed with the word meaning states (see table 4.1).

4.3 WORD SIMILARITY WITH DENSITY MATRICES

One key question in natural language processing is how similar two words, phrases, or sentences are. In vector space models, the similarity of two meaning vectors is measured by their inner product or cosine similarity. This section presents candidates for measuring the similarity of two word meaning density matrices.

4.3.1 Trace Inner Product

The trace inner product is the counterpart to the vector inner product for matrices. Thus, it seems to be an intuitive choice for a similarity measure. It is defined as follows:

$$\langle A, B \rangle = \text{tr}(A^\dagger B) \quad (4.5)$$

Unfortunately, the trace inner product of a matrix with itself does not always equal one. However, for a useful similarity measure, it should be one to indicate that the similarity of a matrix with itself is maximal.

For example the trace inner product of the following completely mixed state with itself does not equal one:

$$\begin{aligned} \rho &= \frac{1}{2}(|0\rangle\langle 0| + |1\rangle\langle 1|) \\ \text{tr}(\rho^\dagger \rho) &= \text{tr}(\rho^2) = \frac{1}{2} \end{aligned}$$

Sometimes the trace inner product of a matrix with itself is even smaller than the trace inner product with another matrix, e.g. for the following states:

$$\begin{aligned}\rho &= \frac{1}{4}|0\rangle\langle 0| + \frac{3}{4}|1\rangle\langle 1| \\ \sigma &= |1\rangle\langle 1|\end{aligned}$$

The trace inner product of ρ with itself is smaller than the trace inner product of ρ and σ :

$$\begin{aligned}\text{tr}(\rho\sigma) &= \frac{3}{4} \\ \text{tr}(\rho^2) &= \frac{5}{8}\end{aligned}$$

This indicates that ρ is more similar to σ than itself. Consequently, even though it appears to be a natural choice, it is not a suitable measure for the similarity of two matrices. [24]

4.3.2 Trace Distance

In quantum mechanics, the trace distance is a commonly used distance measure for density matrices. It is defined as follows:

$$D(\rho, \sigma) = \frac{1}{2}\text{tr}|\rho - \sigma| \quad (4.6)$$

Where $|\rho| = \sqrt{\rho^\dagger\rho}$. The trace distance of a density matrix with itself equals zero, and the trace distance between two density matrices can have a maximum value of one:

$0 \leq D(\rho, \sigma) \leq 1$. Therefore, the following similarity measure can be defined:

$$S(\rho, \sigma) = 1 - D(\rho, \sigma) \quad (4.7)$$

[24]

4.3.3 Fidelity

The fidelity is another widely used distance measure for density matrices in quantum mechanics. It is defined as follows:

$$F(\rho, \sigma) = \text{tr}\sqrt{\rho^{1/2}\sigma\rho^{1/2}} \quad (4.8)$$

The following properties make it a highly suitable measure of similarity:

- Symmetry: $F(\rho, \sigma) = F(\sigma, \rho)$

- $0 \leq F(\rho, \sigma) \leq 1$
- $F(\rho, \sigma) = 1 \Leftrightarrow \rho = \sigma$
- The fidelity of two pure states $|\psi\rangle\langle\psi|$ and $|\phi\rangle\langle\phi|$ is equal to the absolute value of the inner product of their vector representations:

$$F(|\psi\rangle\langle\psi|, |\phi\rangle\langle\phi|) = |\langle\psi|\phi\rangle|$$

[24]

Examples for computing the similarity of density matrices will be provided in the lexical entailment examples below.

4.4 LEXICAL ENTAILMENT WITH DENSITY MATRICES

The natural way in which density matrices embody probability distributions can be used to model lexical entailment. The distributional hypothesis of hyponymy summarizes the phenomenon of lexical entailment very accurately.

Definition 8. [12] **Distributional Hypothesis for Hyponymy**

‘The meaning of a word w subsumes the meaning of a word v if and only if it is appropriate to use w in all the contexts v is used.’

This hypothesis and the meaning of subsumption become more evident when considering a text about dogs in which the word ‘dog’ is blanked out. For each gap in the text, the question is whether the word ‘dog’ or ‘animal’ fits in. Since ‘animal’ subsumes the meaning of the word ‘dog’ and can be used in all cases where ‘dog’ could be used, it is impossible to tell with certainty which of the two words belongs in the gaps.

Next, consider a text about an animal, blank out that word as well, and ask the same question about whether the word ‘dog’ or ‘animal’ fits in the gaps. One sentence in the text says, ‘The ___ has black and white stripes and lives in the savannah.’. In this context, it quickly becomes clear that the word ‘animal’ and not ‘dog’ should certainly be used in the blank. [24]

Whether a word v is subsumed by another word w , is given by the ability of the word w to represent the meaning of v . Vice versa, the question might turn to how distinguishable the words v and w are given some context. In the case of probability distributions or density matrices, distinguishability can be quantified by the relative entropy.

Definition 9. [12] The (quantum) relative entropy of two density matrices ρ and σ is defined as follows:

$$N(\rho||\sigma) := tr(\rho \log \rho) - tr(\rho \log \sigma) \quad (4.9)$$

where $0 \cdot \log(0) = 0$ and $x \cdot \log(0) = \infty$ with $x \neq 0$, by convention

A measure for the representativeness between two density matrices can be grounded on the relative entropy.

Definition 10. [12] The representativeness describes the degree to which ρ can be represented by σ . It is defined as follows:

$$R(\rho, \sigma) := 1/(1 + N(\rho||\sigma)) \quad (4.10)$$

Where $N(\rho||\sigma)$ is the quantum relative entropy.

The following properties of quantum relative entropy and representativeness support this choice for a measure of representativeness:

- $N(\rho||\sigma) > 0$
- $0 \leq R(\rho, \sigma) \leq 1$
- $R(\rho, \sigma) = 1 \Leftrightarrow \rho = \sigma$
- $N(\rho||\sigma) = \infty$ if $supp(\rho) \cap ker(\sigma) \neq 0$, and finite otherwise
- $R(\rho, \sigma) = 0 \Leftrightarrow supp(\rho) \cap ker(\sigma) \neq 0$

Where $supp$ is the support of a matrix $supp(\rho) = \{\vec{v} \in V | \rho\vec{v} \neq 0\}$ and ker is the kernel of a matrix $ker(\rho) = \{\vec{v} \in V | \rho\vec{v} = 0\}$.

In the preceding example, the sentence ‘___ has black and white stripes and lives in the savannah’ renders the word ‘animal’ perfectly distinguishable from the word ‘dog’. Thus, ‘dog’ is not able to represent ‘animal’ and $R(\widehat{animal}, \widehat{dog}) = 0$ and $supp(\widehat{animal}) \cap ker(\widehat{dog}) \neq 0$. Vice versa ‘animal’ is a good representative for ‘dog’, thus $R(\widehat{dog}, \widehat{animal}) > 0$ and $R(\widehat{animal}, \widehat{dog}) \neq 0$ must be fulfilled. ‘Animal’ is accordingly a hyperonym of ‘dog’ and ‘dog’ is a hyponym of animal. Generally, this can be formalized as follows:

$$\begin{aligned} \rho \prec \sigma & \text{ if } R(\rho, \sigma) > 0 \\ \rho \sim \sigma & \text{ if } \rho \prec \sigma \text{ and } \sigma \prec \rho \end{aligned} \quad (4.11)$$

ρ is a hyponym of σ if $\rho \prec \sigma$ and $\rho \approx \sigma$. The value of $R(\rho, \sigma)$ quantifies the degree to which a word is a hyponym of another one. The bigger the value of $R(\rho, \sigma)$, the better ρ is represented by σ .

This quantification of hyponymy between words is also applicable to entire sentences. For example, replacing all words in a transitive sentence with their hyponyms results in the modified sentence being a hyponym of the original one.

Theorem 2. [12] If $\rho, \sigma, \delta, \gamma \in N^* \otimes N$ are nouns, $\alpha, \beta \in N^* \otimes N \otimes S^* \otimes S \otimes N^* \otimes N$ are transitive verbs and ρ, δ and α are hyponyms of σ, γ and β according to equation 4.11, then:

$$S(r)(\rho \otimes \alpha \otimes \delta) \prec S(r)(\sigma \otimes \beta \otimes \gamma) \quad (4.12)$$

Where $S(r)$ is the strong monoidal functor from equation 4.4.

For proof of this theorem, see [12].

Two examples for lexical entailment between nouns and sentences are provided in the following.

Example 18. [12] **Entailment and Similarity between Nouns**

Let the words ‘pub’, ‘pitcher’ and ‘tonic’ span the basis of a noun space N . The words ‘lager’ and ‘ale’ can be expressed in that basis as pure states as follows:

$$\begin{aligned} |lager\rangle &= 6 \cdot |pub\rangle + 5 \cdot |pitcher\rangle + 0 \cdot |tonic\rangle \\ |ale\rangle &= 7 \cdot |pub\rangle + 3 \cdot |pitcher\rangle + 0 \cdot |tonic\rangle \end{aligned}$$

Their density matrix representation is then as follows:

$$\begin{aligned} \widehat{lager} &= |lager\rangle\langle lager| \\ \widehat{ale} &= |ale\rangle\langle ale| \end{aligned}$$

To highlight the different levels of words represented as basis vectors, pure states, and mixed states, these are referred to as basis words, atomic words, and non-atomic words. ‘pub’, ‘pitcher’ and ‘tonic’ are then the basis words and ‘lager’ and ‘ale’ are atomic words. According to the taxonomy, $beer = lager + ale$, the word ‘beer’ is a non-atomic word that subsumes the two words ‘lager’ and ‘ale’ and can be embodied by a mixed state.

The density matrix of non-atomic words is obtained by counting co-occurrences with tuples of the basis words ‘pub’, ‘pitcher’, and ‘lager’ in a window, in which the other basis words do not occur. The co-occurrences C_{ij} are counted for all tuples of basis words $|b_i\rangle\langle b_j|$.

Density matrices of non-atomic words are thus computed as follows:

$$\rho_{non-atomic} = \sum_{ij} C_{ij} |b_i\rangle\langle b_j|$$

If ‘beer’ is observed 6 times with ‘pub’ only, seven times with both ‘pub’ and ‘pitcher’, and never with ‘tonic’, its density matrix has the following form:

$$\begin{aligned} \widehat{beer} &= 6 \cdot |pub\rangle\langle pub| + 7 \cdot (|pub\rangle + |pitcher\rangle)(\langle pub| + \langle pitcher|) \\ &= 13 \cdot |pub\rangle\langle pub| + 7 \cdot |pub\rangle\langle pitcher| + 7 \cdot |pitcher\rangle\langle pub| + 7 \cdot |pitcher\rangle\langle pitcher| \end{aligned}$$

Then all density matrices are normalized by $\frac{\rho}{tr(\rho)}$.

The similarity between ‘beer’ and ‘lager’ can be computed using the fidelity from equation 4.8:

$$F(\widehat{lager}, \widehat{beer}) = tr(\sqrt{\widehat{lager}^{\frac{1}{2}} \cdot \widehat{beer} \cdot \widehat{lager}^{\frac{1}{2}}}) = 0.93$$

As was to be expected, the two words are very similar. The representativeness between ‘lager’ and ‘beer’ is:

$$R(\widehat{lager}, \widehat{beer}) = \frac{1}{1 + tr(\widehat{lager} \cdot \log(\widehat{lager}) - \widehat{lager} \cdot \log(\widehat{beer}))} = 0.82$$

and the representativeness between ‘beer’ and ‘lager’ is

$$R(\widehat{beer}, \widehat{lager}) = 0$$

Thus $\widehat{lager} \prec \widehat{beer}$ and $\widehat{lager} \approx \widehat{beer}$ and ‘lager’ is a hyponym of ‘beer’.

Example 19. [12] Entailment between sentences

Now, the noun space N of the previous example is extended by the basis words ‘patient’, ‘mental’ and ‘surgery’. The words ‘psychiatrist’ and ‘doctor’ can be described by the following density matrices:

$$\begin{aligned} \widehat{psychiatrist} &= 2 \cdot |patient\rangle\langle patient| + 5 \cdot |mental\rangle\langle mental| \\ \widehat{doctor} &= 5 \cdot |patient\rangle\langle patient| + 2 \cdot |mental\rangle\langle mental| + 3 \cdot |surgery\rangle\langle surgery| \end{aligned}$$

The fidelity between ‘psychiatrist’ and ‘doctor’ is:

$$F(\widehat{psychiatrist}, \widehat{doctor}) = 0.76$$

This indicates the two words are very similar. Their representativeness is:

$$R(\widehat{psychiatrist}, \widehat{doctor}) = 0.49 \qquad R(\widehat{doctor}, \widehat{psychiatrist}) = 0$$

Thus, $\widehat{psychiatrist} \prec \widehat{doctor}$ and $\widehat{psychiatrist} \approx \widehat{doctor}$ and ‘psychiatrist’ is a hyponym of ‘doctor’.

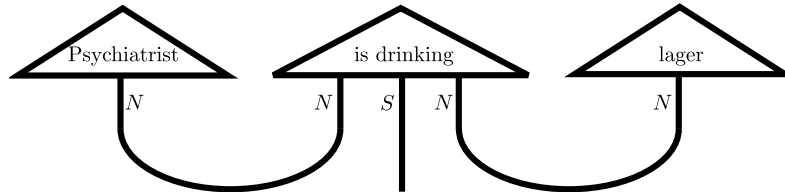
Since transitive verbs have the type $n^r sn^l$, they are tensors in the meaning space $N \otimes S \otimes N$. Thus, they are higher-order tensors than nouns, which are elements of the meaning space N . To obtain a transitive verb’s tensor, a matrix is constructed such that each entry (i, j) reflects the verb’s co-occurrence frequency with a subject related to the i^{th} basis word and an object related to the j^{th} basis word. Afterwards, the outer product of this matrix with itself raises it to the required higher-order tensor in the space of density matrices. In this example, the relevant part of this matrix for the verb ‘drink’ could look like this:

$ drink\rangle$	pub	pitcher	tonic
patient	4	5	3
mental	6	3	2
surgery	1	2	1

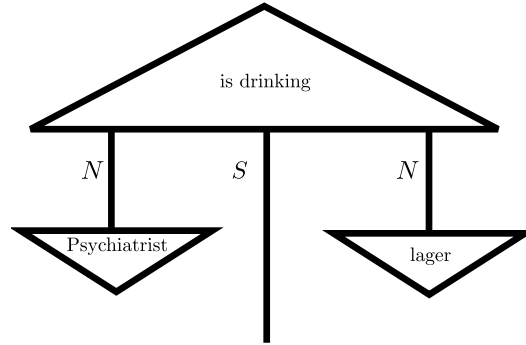
The outer product of which provides the final meaning representation of the verb:

$$\widehat{drink} = |drink\rangle\langle drink|$$

With these components the sentences ‘Psychiatrist is drinking lager’ and ‘Doctor is drinking beer’ can be examined for their hyponymy relationship. Diagrammatically the sentence ‘Psychiatrist is drinking lager’ is depicted as follows:



Using the swing rule from 3.2 b, the subject’s and object’s triangles can be swung around, resulting in the following diagram:



The diagram of ‘Doctor is drinking beer’ is visualized analogously to the graphical representation above. Therefore, these sentences’ density matrices are calculated as follows:

$$\begin{aligned} \widehat{\text{Psychiatrist is drinking lager}} &= \widehat{\text{drink}} \circ (\widehat{\text{psychiatrist}} \otimes \widehat{\text{lager}}) \\ \widehat{\text{Doctor is drinking beer}} &= \widehat{\text{drink}} \circ (\widehat{\text{doctor}} \otimes \widehat{\text{beer}}) \end{aligned}$$

The fidelity and representativeness for the two sentences are as follows:

$$\begin{aligned} F(\widehat{\text{Psychiatrist is drinking beer}}, \widehat{\text{Doctor is drinking beer}}) &= 0.81 \\ R(\widehat{\text{Psychiatrist is drinking beer}}, \widehat{\text{Doctor is drinking beer}}) &= 0.53 \\ R(\widehat{\text{Doctor is drinking beer}}, \widehat{\text{Psychiatrist is drinking beer}}) &= 0 \end{aligned}$$

The high fidelity indicates that the two sentences are very similar and from the representativeness follows:

$$\begin{aligned} \widehat{\text{Psychiatrist is drinking lager}} &\prec \widehat{\text{Doctor is drinking beer}} \\ \widehat{\text{Psychiatrist is drinking lager}} &\approx \widehat{\text{Doctor is drinking beer}} \end{aligned}$$

Thus, the sentence ‘Psychiatrist is drinking lager’ is a hyponym of the sentence ‘Doctor is drinking beer’. This emphasizes the statement from theorem 2 that replacing words (‘doctor’, ‘beer’) in a sentence by their hyponyms (‘psychiatrist’, ‘lager’) results in the new sentence being a hyponym of the original one.

4.5 DENSITY MATRICES FOR WORD AMBIGUITY

Word ambiguity is another phenomenon where the ability of density matrices to capture probability distributions is useful. The information content of a probability distribution is measured by its entropy. With word meaning density matrices, the information content translates directly into a measure of word ambiguity. In quantum mechanics, the

information content of a density matrix is measured by the von Neumann entropy. [11]

Definition 11. [11] **Von Neumann Entropy**

For a density matrix ρ with eigendecomposition $\rho = \sum_i \lambda_i |e_i\rangle\langle e_i|$, where $|e_i\rangle$ are the eigenvectors and λ_i are the eigenvalues of ρ , the von Neumann Entropy is defined as follows:

$$S(\rho) = -\text{tr}(\rho \ln \rho) = -\sum_i \lambda_i \ln \lambda_i \quad (4.13)$$

A word is highly ambiguous if its density matrix has a large von Neumann entropy. As a word's context can provide more information about its meaning, the word can be disambiguated by composing it into phrases. The von Neumann entropy of the phrase or sentence is then smaller than that of the individual word. For the quantum states described by these density matrices, this means that the phrase's state is less mixed or purer than that of the single word. Therefore, word meaning disambiguation corresponds to a purification of the word's quantum state.

Example 20. Let the noun space N be spanned by the two basis words 'metal' and 'horn'. The word 'nail' can either describe a piece of metal for hammering into a wall or a structure made of horn growing at a finger's tip. Before composition with another word, the word 'nail' is described by the following density matrix:

$$\widehat{nail} = 0.5|metal\rangle\langle metal| + 0.5|horn\rangle\langle horn|$$

The von Neumann entropy of which is as follows:

$$S(\widehat{nail}) = 0.69$$

After composition with the word 'rusty', it is clear that the piece of metal is meant and not the part of a finger. Thus, the density matrix nears the pure state $|metal\rangle\langle metal|$:

$$\widehat{rusty\ nail} = 0.9|metal\rangle\langle metal| + 0.1|horn\rangle\langle horn|$$

The von Neumann entropy of this phrase is then:

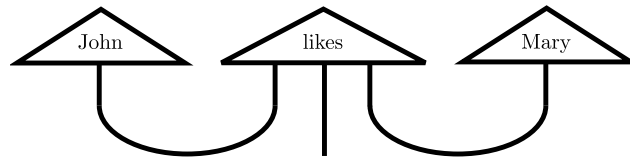
$$S(\widehat{rusty\ nail}) = 0.33$$

It becomes evident that the phrase 'rusty nail' is significantly less ambiguous than the word 'nail'.

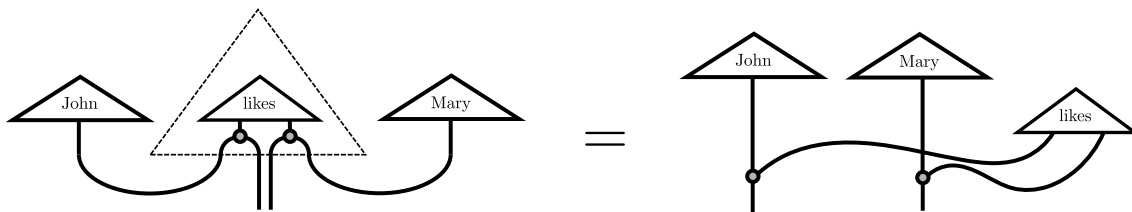
In [11] it has been shown that the von Neumann entropy of individual ambiguous words is larger than after composition with other words, which provide context that should disambiguate the word's meaning. In general, the states after composition and disambiguation are not entirely pure but less mixed.

4.6 MEANING UPDATING OF DENSITY MATRICES

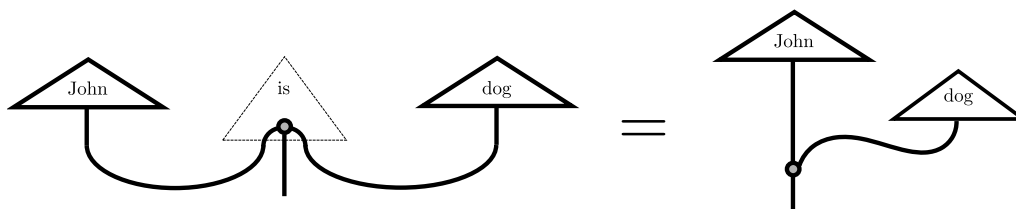
In order to calculate sentence meanings, investigate entailment relationships between phrases, or study the disambiguation abilities of context words, words need to be composed into phrases and sentences. So far, this was done by calculating the tensor product of the word meaning vectors or density matrices and applying the reduction maps that originate from the sentence's grammatical structure. However, other interesting composition methods exist for density matrices. These can be introduced using the diagrammatic calculus. So far, diagrams of transitive sentences looked like this:



However, other possible representations for different types of words were introduced within the DisCoCat model. In [25] the following form of transitive verbs was proposed:



For the purposes of this section, the grey dot can be thought of as a placeholder for an unknown arbitrary operation, which composes the words' meaning representations. To demonstrate this with an example, a simpler phrase is considered. The verb 'be' is a so-called 'linking' verb. Linking verbs connect a subject with an object that provides information about the subject. Using the new graphical notation, the diagram of the sentence 'John is a dog', can be depicted as follows:



Now, different operations can be considered as candidates for the grey dot. The simplest one, matrix multiplication, is not an option since the result is not a density matrix. Moreover, elementwise multiplication is commutative and thus does not respect the words' order, which disqualifies it for sentences with grammatical structure. Another candidate originates from the Birkhoff-von Neumann logic [26]. It states that 'any proposition about a physical system corresponds to a subspace A , or its corresponding projector P_A ' [27]. In natural language, a proposition is the meaning of a declarative sentence. Following this argument [28, 29], propositions like 'being a dog' can be considered projectors. Thus the proposition 'being a dog' can be imposed onto 'John' by a projector as follows:

$$P_{dog} \circ John \circ P_{dog}$$

The resulting matrix of such a term will not be normalized. Therefore, the term density matrix will also be used for sub-normalized and super-normalized positive matrices in the following. If 'John' was a density matrix and 'dog' was a projector, this would be the new composition method. But 'dog' is not a projector but a density matrix as well. Fortunately, density matrices can be represented as follows as weighted sums over projectors in the form of their eigendecomposition:

$$\rho = \sum_i x_i P_i \tag{4.14}$$

Based on this, the following two composition methods have been proposed:

$$\begin{aligned} \rho \square \sigma &:= \sum_i x_i (P_i \circ \rho \circ P_i) & \text{with } \sigma &:= \sum_i x_i P_i \\ \rho \Delta \sigma &:= \left(\sum_i x_i P_i \right) \circ \rho \circ \left(\sum_j x_j P_j \right) & \text{with } \sigma &:= \sum_i x_i^2 P_i \end{aligned} \tag{4.15}$$

The first one, denoted by \square , was introduced in [30, 31] and is called 'fuzz'. The second one was initially introduced in quantum theory for bayesian inference [32, 33, 34] and is called 'phaser'. Since the results of these operations are not necessarily normalized, they need to be re-normalized after each update. Neither the fuzz nor the phaser are completely

positive maps and thus no morphisms in the category $\mathbf{CPM}(\mathbf{FHilb})$. However, especially the phaser showed promising results as a composition method in practice [20, 13]. Since the DiCoCat formalism claims to have a unique update mechanism and work entirely within the framework of compact closed monoidal categories, the authors of [27] find another meaning category of ‘doubled’ density matrices in which the composition methods of the fuzz and phaser can be unified such that the new composition method is a morphism within that new category.

Presenting this unification in detail goes beyond the scope of this thesis. The results of the provided implementation of quantum density matrix word embeddings are not evaluated on tasks where words need to be composed. These methods are introduced for completeness only, as they were used in the paper, this thesis is based on [13]. Further work would have to use these methods to be comparable.

4.7 LEARNING DENSITY MATRICES CLASSICALLY

So far, only the theoretical concept of constructing word embeddings, based on co-occurrence frequencies with context words, was considered. In the case of density matrices, the probability distributions over underlying meanings were built manually based on taxonomies like $beer = lager + ale$.

In practice, these embeddings are learned using machine learning models. In [13] the artificial neural network architecture Word2DM was introduced to learn density matrix word embeddings. It is an adaption of the widely used artificial neural network architecture Word2Vec that learns word meaning vectors [2]. To understand the Word2DM model, the concept of Word2Vec is introduced first. There are two versions of the Word2Vec algorithm. Both are trained on tasks, which cause the network’s weights to reflect the co-occurrence statistics of words in the text. Thus, the weights can be used as word embedding vectors. The first of the two algorithms is called Continuous-Bag-of-words. It is trained to predict the occurrence of a word based on context words within a window around the word itself. The second version of the Word2Vec algorithm is called Continuous Skip-gram. This version is trained to predict the context words of a word within a specific range based on the word itself.

The Word2DM model [13] is an adaption of the Continuous Skip-gram model that learns density matrices as word embeddings instead of vectors. It forms the basis for the quantum density matrix word embedding model that will be proposed in the next chapter.

This section introduces the Word2Vec Skip-gram model and the modifications within the Word2DM model.

4.7.1 Word2Vec Continuous Skip-gram

This section explains the concept behind the Word2Vec Skip-gram model with negative sampling, based on the online tutorial [35]. The goal is to learn word embedding vectors from a text corpus. Therefore a neural network is trained on a particular task, and the weights of the trained neural network provide the actual word embeddings. The sentences in a text corpus are processed one by one. A word in the sentence is chosen as the target word and another word that occurs within a window around the target word is selected as a context word. Figure 4.1 illustrates the procedure for finding target-context word pairs with window size two. The context words are picked from the two words before and after the target word.

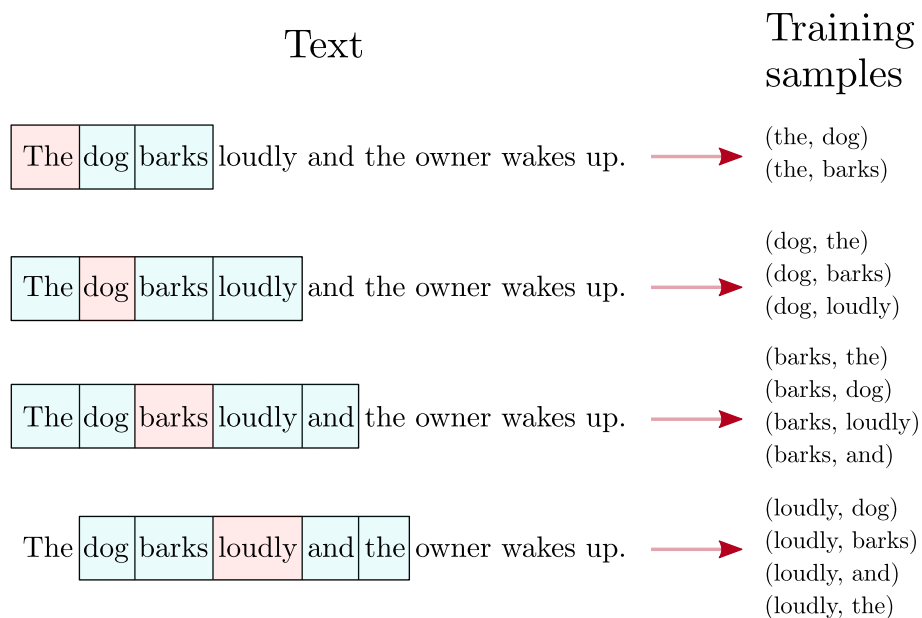


Figure 4.1: Procedure for finding target-context word pairs with window size two. Context words are selected within a window size 2 before and after the target word.

The neural network is then trained as a classifier with the target words as input and the context words as labels that shall be predicted. The output is a probability distribution over all words in the vocabulary of being the chosen context word, corresponding to the given target word.

Figure 4.2 illustrates the network's architecture. The target word is encoded into a one-hot vector of length V , representing the vocabulary size, and fed into the neural network as input. The network has one linear hidden layer and a linear output layer with a softmax activation function. The output is a vector of length V , representing the probability distribution over

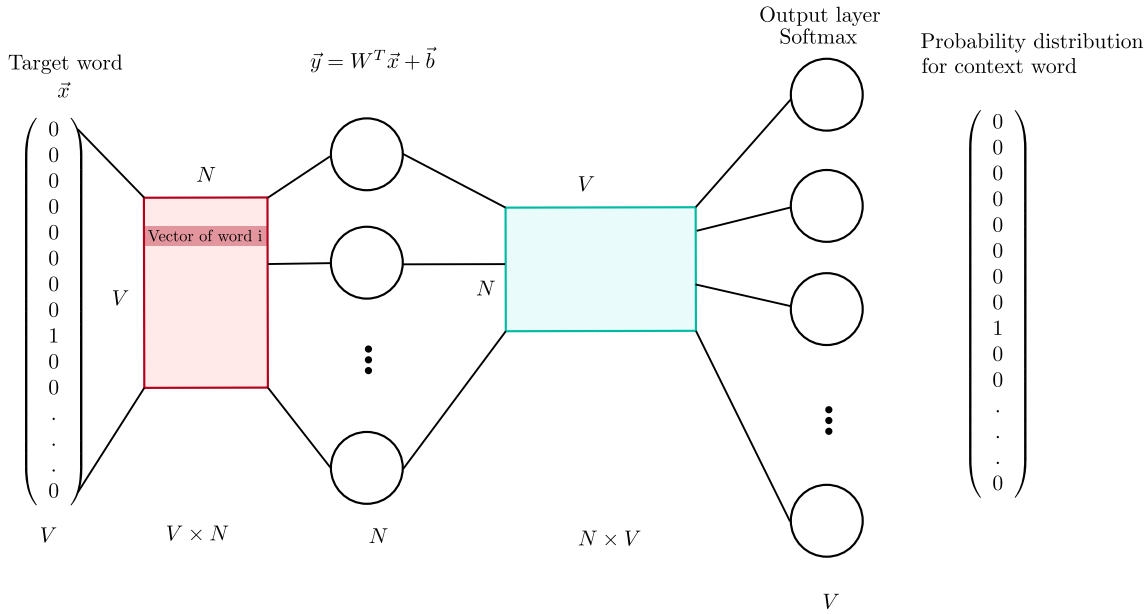


Figure 4.2: Skip-gram architecture. The input is the target word’s one-hot vector. The network has one linear hidden layer and a linear output layer with a softmax activation function. The output is a probability distribution over all words in the vocabulary of being the context word corresponding to the given target word. The loss between the output and the true context word’s one-hot vector is computed to update the network’s weights. The rows of the hidden layer’s weight matrix provide the word embedding vectors after training.

all words being the context word corresponding to the current word pair. The loss between this output vector and the one-hot vector of the true context word is computed to update the neural network’s weights. Feeding the network all target-context word pairs occurring in the text teaches it the words’ co-occurrence statistics. After training, the network is no longer used for the task it is initially trained on, but instead, the rows of the weight matrix of the hidden layer provide the words’ embedding vectors. This is possible because multiplying the target word vector with this weight matrix returns the row of the weight matrix corresponding to the input as a result of the hidden layer. As the weight matrix has the dimension $V \times N$, the number of neurons in the hidden layer N determines the embedding size.

This neural network architecture embodies the aforementioned idea of John Rupert Firth that ‘you shall know a word by the company it keeps’. Words that occur together are trained to have similar word embedding vectors.

A drawback is that the weight matrices become very large for extensive vocabulary sizes. For example, a vocabulary size $V = 10,000$ and an embedding size $N = 300$, result in

three million weights, both in the hidden and the output layer. Training such a network is computationally infeasible. Additional routines on top of the architecture reduce the consumption of computational resources. [35]

Subsampling

The first such routine is called subsampling. By subsampling very frequently occurring words, the number of training samples is reduced. In the example from figure 4.1, even though the word ‘the’ does not tell much about the meaning of the word ‘dog’, it will appear very frequently in its context and in the context of every other noun in the text. Thus, there will be many more samples for the word ‘the’ than are needed to learn its embedding vector. Subsampling introduces a probability for words in the text to be deleted that is proportional to their occurrence frequency. Not all instances of a word are deleted at once, but each instance of the word is kept or deleted depending on that probability. This results in fewer instances of very frequently occurring words in the text.

The probability for keeping a word is given by:

$$P(w_i) = \left(\frac{\sqrt{z(w_i)}}{b} + 1 \right) \cdot \frac{b}{z(w_i)} \quad (4.16)$$

Where w_i is the respective word and $z(w_i)$ is the occurrence frequency of that word in the corpus. A sampling parameter b controls how big or small the probability of keeping a word is. The smaller it is, the more likely words are deleted. Deleting specific instances of words from the text results in deleting the training samples, including them, and thus reduces the number of overall training samples to economize computational resources. [35]

Negative Sampling

Another routine to reduce the consumption of computational resources is negative sampling. As was seen above, the number of weights in the neural network can become very large. In general, the network’s output is trained to be a vector with only a single one at the position corresponding to the correct context word and zeros in all the other positions. Therefore, all weights in the output layer need to be updated in every training step, which is infeasible for large numbers of weights. With negative sampling, not all other positions are trained to be zero in every step, but instead, only a small number of zero-valued positions are chosen for training in each step. Hence, only the weights contributing to the selected positions are updated.

For example, with five negative samples per training step and $300 \times 10.000 = 3$ million weights in the output layer, only the weights corresponding to the five negative samples

and the one positive sample would be updated. Thus, only the weights for six of the output neurons would be updated instead of all. This means only $300 \times 6 = 1800$ weights in the output layer are updated in each training step instead of 3 million.

The negative samples are picked randomly from the text corpus. Therefore, words with more occurrences in the text are chosen more often as negative samples. The probability for a word being selected as a negative sample is given by:

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^n (f(w_j))^{3/4}} \quad (4.17)$$

Where $f(w_i)$ is the absolute number of occurrences of the word w_i in the text corpus. Assigning these frequencies the power of $3/4$ was found to work well because it tends to choose less frequently occurring words a bit more often as negative samples and more frequently occurring words a bit less often. [2]

This routine reduces the number of updated weights in the output layer. For the weights in the hidden layer, no such procedure is needed. Due to the one-hot encoded input, only a small fraction of the weights corresponding to the input word's embedding vector are updated in the hidden layer.

Subsampling and negative sampling reduce the use of computational resources and even provide higher quality word embedding vectors. [2]

Essentially, the skip-gram model with negative sampling optimizes the following objective function:

$$L(\theta) = \log \sigma(v_t^T v_c) + \sum_{k=1}^K \log \sigma(-v_t^T v_{n_k}) \quad (4.18)$$

Where v_t is the embedding vector of the target word, v_c is the embedding vector of the context word, and v_{n_k} are the embedding vectors of the K negative samples. Subsampling happens within the text's pre-processing.

4.7.2 Word2DM

The Word2DM model [13] adapts the skip-gram model with negative sampling to learn density matrices as word embeddings instead of vectors. Therefore the vectors in the objective function from equation 4.18 are replaced by density matrices.

$$L(\theta) = \log \sigma(\text{tr}(A_t A_c)) + \sum_{k=1}^K \log \sigma(-\text{tr}(A_t A_{n_k})) \quad (4.19)$$

Where A_t is the density matrix of the target word, A_c is the density matrix of the context word, and A_{n_k} are the density matrices of the K negative samples.

Without further adjustment, the matrices A are just arbitrary. To be useful for experiments with word ambiguity and lexical entailment, they need to be density matrices that encode probability distributions. Therefore, the properties of density matrices, such as hermiticity, positive semi-definiteness, and unit trace (see section 2.6), need to be enforced. For this purpose, intermediary matrices B are learned, and the actual embedding density matrices are computed as $A = BB^\top$, because the product BB^\top is positive semi-definite for any matrix B .

Including these intermediary matrices in the training procedure results in $3(k + 1)$ matrix multiplications that must be carried out in each training step. By optimizing the form of the objective function, the authors of [13] manage to reduce this number to $K + 1$ matrix multiplications per training step. However, this still increases the consumption of computational resources compared to the very efficient original skip-gram model. While this would still be acceptable, unfortunately, the use of these intermediary matrices results in sub-optimal training updates. Especially when the density matrices of positive target-context samples are very dissimilar before training, the form of the gradients causes the updates to be so small that the target and context words' density matrices do not become significantly more similar. [13]

5

Quantum Density Matrix Word Embeddings

This thesis proposes a hybrid quantum-classical algorithm for learning density matrix word embeddings. Since density matrices describe quantum states, the idea of extracting them from quantum hardware seems obvious. By doing so, the use of intermediate matrices becomes obsolete. The density matrices are extracted at the end of a variational quantum circuit that is capable of learning multi-qubit mixed states. The circuit's parameters are optimized, such that the qubits are prepared into a state corresponding to the density matrix representation of a word's meaning. A separate set of circuit parameters is learned for each word. The parameters are updated using the objective function of the Word2DM model from equation 4.19.

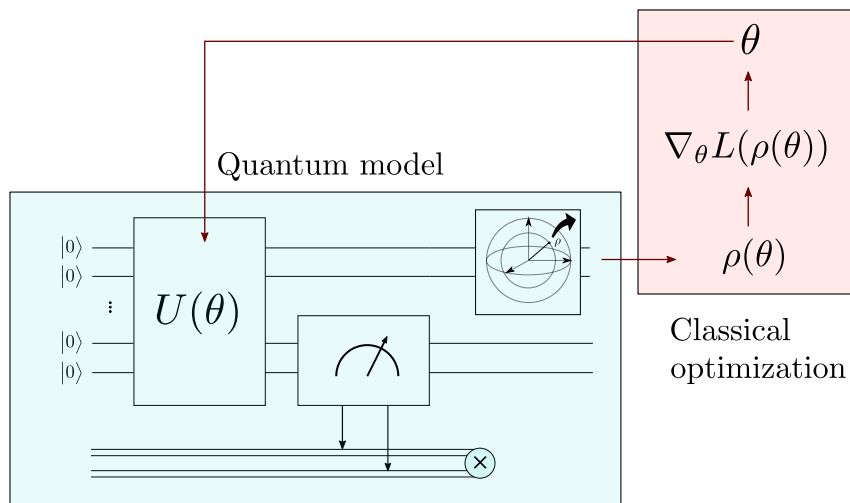


Figure 5.1: Architecture for learning quantum density matrix word embeddings. A parameterized unitary is applied to all qubits. Afterwards a sub-system is traced out in order to put the other qubits into a mixed state. The remaining qubits' state is extracted at the end of the circuit and the parameters are optimized according to equation 4.19.

Figure 5.1 illustrates the model's architecture. $U(\theta)$ is a unitary containing all gate operations in the circuit. After applying the unitary, a sub-system of the circuit is traced out by measuring some qubits and discarding the result. This procedure allows placing

the remaining qubits in a mixed state. The box with the small Bloch Sphere on the upper qubits indicates that their state is extracted from the circuit.

In each training step, the circuit is executed once with the target word's parameters θ_t , once with the context word's parameters θ_c , and K times for the K negative samples with their respective parameters θ_{n_k} . The extracted density matrices are then used to update the parameters according to equation 4.19.

In the following sections, the necessary number of qubits is first discussed. Subsequently, the key points to consider when choosing a circuit ansatz for $U(\theta)$ are introduced. Further, the state's extraction from the circuit and the gradient calculation for the model are discussed. Ultimately, the choices for the first implementation are presented.

5.1 NUMBER OF QUBITS

$N \times N$ dimensional density matrix embeddings can be learned by extracting the state of $\log_2(N)$ qubits. Current quantum computing models are based on manipulating pure states using unitary gates. Therefore, preparing a mixed state is not trivial. Section 2.6 already mentioned that measuring a sub-system of an entangled multi-qubit state also influences the unmeasured qubits' state. Discarding the measurement result instead of revealing it leaves the observer with uncertain knowledge about the state of the non-measured qubits. Thus, these qubits find themselves in a mixed state. Hence, for the preparation of a mixed state, not only $\log_2(N)$, but M additional qubits are required for learning $N \times N$ dimensional word embedding density matrices. No definite indication of the number M to be selected could be identified in the literature.

5.2 CHOICE OF CIRCUIT ANSATZ

From the necessity of additional qubits that have to be entangled with the qubits whose state is extracted at the end of the circuit, follows that the ansatz for $U(\theta)$ has to contain entangling layers.

Further, the optimal ansatz for this task would be able to prepare the qubits into an arbitrary n -qubit state to provide maximal flexibility for the representation of the word meanings. Such an ansatz would necessitate a lot of gates and thus be very deep. Currently, only noisy intermediate-scale quantum computers (NISQ devices) exist whose states are not stable and therefore decay over short periods of time. The circuit's results would consequently be highly corrupted. When using simulators, computational resources are also limited. Therefore, this kind of ansatz is not feasible for implementation, and a useful

trade-off between flexibility and complexity must be found.

A suitable circuit should not be too deep but still, cover enough of the state space so that the learned states reflect the different meanings of the words in the text corpus.

In summary, the following key points have to be considered when choosing an ansatz for $U(\theta)$:

- feasible circuit depth
- enough flexibility to reflect word meanings in the given text corpus
- entanglement with additional qubits to allow for mixed state

5.3 EXTRACTION OF DENSITY MATRICES FROM THE CIRCUIT

The biggest challenge for the implementation on real quantum hardware is the extraction of the state's density matrix at the end of the quantum circuit. Since measuring a quantum system causes its state to collapse into one of the basis states, the information on superposition and mixture of the state is not observable. On real quantum hardware, a system's full state can be reconstructed using a method called quantum state tomography. With simulators, on the other hand, the state is available at any time during the simulation.

5.3.1 Quantum State Tomography

The state of a quantum system can be reconstructed by carrying out different measurement runs on many copies of the state. Therefore, the measurement operators have to build an operator basis in the system's Hilbert space. The state can then be reconstructed from the measurement statistics.

The objective function in equation 4.19 depends on the density matrices of the target word, the context word, and the K negative samples. All these states would have to be reconstructed using quantum state tomography. Thus, on real quantum hardware, the circuit would have to be executed $T(2 + K)$ times in each training step, where T is the number of measurement runs needed for each state reconstruction.

5.3.2 Simulators

The underlying density matrix of the circuit can easily be extracted on simulators, as the system's current underlying state is tracked throughout the entire simulation. Therefore, the circuit must be executed only $(2+K)$ times on a simulator.

5.4 GRADIENTS

Section 2.8.1 addressed challenges regarding the computation of gradients for variational quantum circuits on simulators and real quantum hardware. As this model extracts the density matrix representation of a sub-system as a result of the variational circuit, its mathematical expression is different from that in equation 2.18.

The model's circuit consists of a composite system of two sub-systems A and B with the Hilbert spaces H_A and H_B . The qubits whose state is extracted belong to sub-system A, while the qubits that are traced out belong to sub-system B. The state of the full system is then $|\psi\rangle \in H_A \otimes H_B$.

Initially, the entire system is in state $|0\dots 0\rangle$. Applying the unitary $U(\theta)$ puts the system into the state $|\psi(\theta)\rangle = U(\theta)|0\dots 0\rangle$. Since unitaries preserve the purity of states, its density matrix can be written as $\rho_S(\theta) = |\psi(\theta)\rangle\langle\psi(\theta)|$. To put subsystem A in a mixed state, sub-system B is traced out. This corresponds to taking the partial trace of $\rho_S(\theta)$ over the basis of sub-system B.

$$\rho_A(\theta) = Tr_B(\rho_S(\theta)) := \sum_j^{N_B} (I_A \otimes \langle j|_B)(|\psi(\theta)\rangle\langle\psi(\theta)|)(I_A \otimes |j\rangle_B) \quad (5.1)$$

Where N_B is the dimension of Hilbert space H_B , I_A is the identity operator in H_A and $|j\rangle_B$ are the basis states of H_B .

The partial derivatives $\nabla_{\theta}\rho(\theta)$ of this expression have to be computed for updating the circuit parameters. While the expectation value of an observable B varies smoothly with small variations in the parameter values, this is not the case for the extracted density matrix representation of the underlying state. Thus, on real quantum hardware, the gradients of this expression cannot be estimated using the parameter-shift rule. Finding a procedure to compute the gradients for a density matrix of a sub-system is still a topic of ongoing research, and no common strategy could be identified in the literature.

As discussed in section 2.8.1, the situation is different on simulators. Here, the gradients can be computed analytically.

5.5 IMPLEMENTATION

The model was implemented using a simulator. Thus, density matrices can be extracted, and gradients are analytically computable.

This work did not focus on finding the most suitable circuit ansatz for the given task but rather on testing the concept of the approach. The BasicEntanglerLayer ansatz is a

commonly used ansatz for quantum machine learning models that is provided as a template in the quantum computing Python package PennyLane. One layer of this ansatz is illustrated in figure 5.2. Initially, parameterized X-rotation gates are applied to all qubits, followed by a ring of CNOT gates that induce entanglement. Several of these layers can be executed in succession to increase the circuit's complexity.

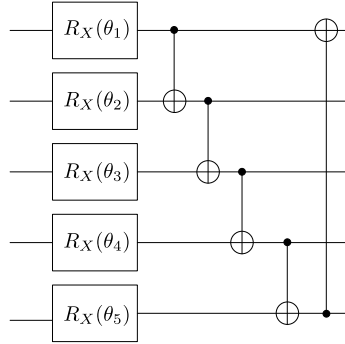


Figure 5.2: A single layer of the BasicEntanglerLayer ansatz. Parameterized X-rotation gates are applied to all qubits, followed by a ring of CNOT gates.

In the first iteration, a single BasicEntanglerLayer is used on a total of five qubits. Three of them are traced out, and the state of the remaining two is extracted at the end of the circuit, yielding 4×4 -dimensional density matrix word embeddings. Figure 5.3 illustrates the entire circuit diagram used in the implementation.

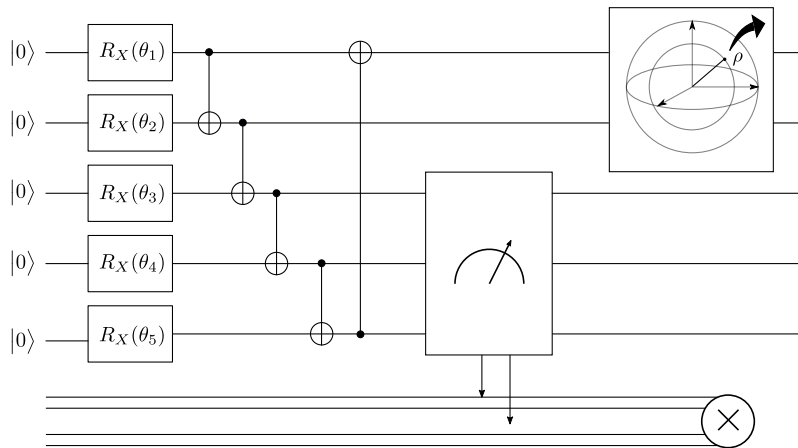


Figure 5.3: Full circuit diagram of the implemented model with a single BasicEntanglerLayer. First, the operations of BasicEntanglerLayer are applied to all five qubits. Afterwards three qubits are traced out, and the density matrix representation of the other two qubits' state is extracted at the end of the circuit.

The same setup is used in a second iteration, but with two BasicEntanglerLayers. In each

iteration, the model is trained in 10 independent training runs of 100 epochs each. The implementation uses parts of the code provided with the Word2DM model [13] to preprocess the training data and evaluate the similarity of the learned density matrix word embeddings.

5.5.1 Choice of Package

The two quantum computing Python packages Qiskit and PennyLane, were considered and investigated for implementation. Both provide an intuitive approach to composing quantum circuits and an interface for parameter optimization with PyTorch. Moreover, PennyLane has automatic differentiation implemented for circuits with density matrices as output when using simulators. This was the decisive factor for realizing the implementation in PennyLane. This section provides a short overview of the two libraries.

Qiskit

Qiskit was first considered for implementation. Variational quantum circuits can be easily implemented in Qiskit. It offers several circuit ansatz templates and the TorchConnector, allowing variational quantum circuits to be used as layers in PyTorch. Thus, quantum layers can be combined with classical PyTorch layers to build hybrid quantum-classical algorithms. The model parameters can then be updated using PyTorch's automatic differentiation. In order to be interfaced via the TorchConnector, the circuit has to be implemented as an instance of the NeuralNetwork class of Qiskit. In this class, the forward and backward methods for the model have to be specified. The forward method takes instances of the circuit parameters as input, executes the circuit, and returns the circuit's output. The backward method computes the gradients of the circuit parameters for optimization. Qiskit provides several pre-implemented NeuralNetwork instances that can be used as templates for building quantum models. These are based on the evaluation of quantum mechanical observables and the measurement statistics of a circuit, but none of them implement a circuit described by equation 5.1. Thus, a custom instance of the NeuralNetwork class would have to be implemented according to equation 5.1, with its gradient computation in the backward method.

PennyLane

PennyLane has a stronger focus on the implementation of quantum machine learning models. The PyTorch interfacing and the gradient computation are more advanced than in Qiskit, which improves the usability. Furthermore, automatic differentiation for circuits of the form

of equation 5.1 is pre-implemented for simulators, making PennyLane the obvious choice for implementing the proposed model.

Variational circuits are implemented as instances of the `qnode` class. The circuit's parameters can be implemented as PyTorch variables and optimized using PyTorch's training procedure. The desired gradient calculation method and the device on which the circuit is to be executed, whether real quantum computer or simulator, can be set as parameters of the `qnode`. Compared to Qiskit, PennyLane provides a broader range of circuit ansatz templates.

5.6 EVALUATION - CHOICE OF SIMILARITY MEASURE

The used objective function of the Word2DM model (equation 4.19) aims to increase the similarity of co-occurring words and decrease the similarity of those that do not occur together. Thus, the model's training success can be measured by evaluating the word similarities between the learned meaning density matrices.

Section 4.3 discussed different possible candidates for a similarity measure with their respective advantages and disadvantages. It was found that the trace inner product is not a good choice to measure the similarity of two density matrices.

Nevertheless, it is used in the implemented model to evaluate the learned density matrices since the objective function of the Word2DM model is based on the trace inner product.

In order to use a different similarity measure for evaluation, the model would have to be trained using another objective function, designed accordingly.

5.7 DATA

This section describes the different datasets used for training. For the first validation, a dummy dataset was developed. Afterwards a slightly bigger dataset with actual words was used.

5.7.1 ABC Dataset

The first dataset will be referred to as the ABC dataset. It consists of the following three 'sentences' with three dummy words respectively:

```
a1 a2 a3
b1 b2 b3
c1 c2 c3
```

For training with this dataset, context words were chosen within a window size of two. No subsampling was used, and one negative sample was considered in each training step. After training, the a_i should be similar, the b_i should be similar, and the c_i should be similar while a_i should not be similar to the b_i and c_i and also the b_i and c_i should not be similar.

5.7.2 Animal Dataset

The second dataset will be referred to as the animal dataset. It consists of the following nine sentences with a total vocabulary size of 25.

```
dogs enjoy cuddling
cats prefer being alone
dogs depend on humans
cats are independent
dogs bark
cats meow
cats and dogs live in houses
lions belong into the wild
lions hunt antelopes
```

For training with this dataset, context words were selected within a window size of two. No subsampling was used, and one negative sample was considered in each training step. Ultimately after training, the words occurring within a sentence should be more similar than those not occurring within the same sentence.

6

Results and Discussion

In a functional learning model, density matrix embeddings of co-occurring words should be more similar than those that do not occur together. In the following, co-occurring words are referred to as ii-word pairs, and words that do not occur together are referred to as ij-word pairs. ii-word pairs should ideally be highly similar, while ij-word pairs should not be similar.

The following sections present the results for the different datasets and circuit ansaetze.

6.1 ABC DATASET

6.1.1 Single BasicEntanglerLayer

Table 6.1 shows the average word similarities of the first training iteration on the ABC dataset with a single BasicEntanglerLayer. The average word similarities of the ii- and ij-word pairs were computed for each run and overall.

run	ii similarity	ij similarity
1	0.78	0.00
2	0.99	0.00
3	0.69	0.14
4	0.63	0.12
5	0.66	0.15
6	0.78	0.00
7	0.78	0.41
8	0.83	0.17
9	0.45	0.13
10	0.50	0.13
overall average	0.71	0.08

Table 6.1: Average word similarity of density matrix word embeddings on the ABC dataset with a single BasicEntanglerLayer

These results demonstrate that the density matrices learned by the proposed model can reflect the word similarities within the ABC-corpus. The trace inner product between

density matrices of ii-word pairs becomes very large, on average 0.71, while that between density matrices of ij-word pairs becomes very small, on average 0.08. The difference between the average similarity of the ii- and ij-word pairs is $0.71 - 0.08 = 0.63$. This indicates a clear distinguishability. Furthermore, these 4×4 density matrices were learned with only five model parameters, which is considerably less than the 16 model parameters needed in the Word2DM model.

Table 6.2 shows the detailed results for the second run of this iteration as an example. The second run had the most significant difference between the similarities of ii- and ij-word pairs. The average trace inner product of ii-word pairs is 0.99 and that of ij-word pairs smaller than 10^{-5} .

word 1	word 2	similarity	word 1	word 2	similarity
a1	a2	0.99	a1	b1	$1.93 \cdot 10^{-9}$
a1	a3	0.99	a1	b2	$2.27 \cdot 10^{-9}$
a2	a3	0.99	a1	b3	$1.17 \cdot 10^{-9}$
b1	b2	0.99	a2	b1	$1.56 \cdot 10^{-9}$
b1	b3	0.99	a2	b2	$1.89 \cdot 10^{-9}$
b2	b3	0.99	a2	b3	$8.02 \cdot 10^{-10}$
c1	c2	0.99	a3	b1	$1.53 \cdot 10^{-8}$
c1	c3	0.99	a3	b2	$1.56 \cdot 10^{-8}$
c2	c3	0.99	a3	b3	$1.45 \cdot 10^{-8}$

word 1	word 2	similarity	word 1	word 2	similarity
a1	c1	$2.03 \cdot 10^{-5}$	b1	c1	$7.25 \cdot 10^{-6}$
a1	c2	$1.62 \cdot 10^{-6}$	b1	c2	$3.47 \cdot 10^{-7}$
a1	c3	$9.53 \cdot 10^{-7}$	b1	c3	$5.06 \cdot 10^{-7}$
a2	c1	$2.05 \cdot 10^{-5}$	b2	c1	$7.06 \cdot 10^{-6}$
a2	c2	$1.80 \cdot 10^{-6}$	b2	c2	$1.63 \cdot 10^{-7}$
a2	c3	$1.13 \cdot 10^{-6}$	b2	c3	$3.22 \cdot 10^{-7}$
a3	c1	$2.18 \cdot 10^{-5}$	b3	c1	$1.20 \cdot 10^{-5}$
a3	c2	$3.12 \cdot 10^{-6}$	b3	c2	$5.12 \cdot 10^{-6}$
a3	c3	$2.45 \cdot 10^{-6}$	b3	c3	$5.28 \cdot 10^{-6}$

Table 6.2: Detailed word similarities of density matrix embeddings of the best run on the ABC dataset with a single BasicEntanglerLayer

6.1.2 Two BasicEntanglerLayers

Table 6.3 shows the results of an iteration with two BasicEntanglerLayers on the ABC dataset. Compared to the single BasicEntanglerLayer, the ii-word pairs are less similar, and

the ij-word pairs are more similar. These results suggest that adding layers increases the complexity but not the model’s flexibility. This might be because the BasicEntanglerlayer only contains rotations around the X-axis. Adding layers does not add new degrees of freedom but only complexity to the rotations around one axis.

run	ii similarity	ij similarity
1	0.56	0.48
2	0.87	0.13
3	0.51	0.28
4	0.70	0.43
5	0.70	0.30
6	0.70	0.39
7	0.80	0.30
8	0.77	0.49
9	0.62	0.37
10	0.80	0.32
overall average	0.70	0.35

Table 6.3: Average word similarity of density matrix word embeddings on the ABC dataset with two BasicEntanglerLayers

6.2 ANIMAL DATASET

After the model showed the expected results for the ABC dataset, the larger animal dataset was used. Table 6.4 shows the corresponding results. A significantly higher similarity could be observed among the ii-word pairs than among the-ij word pairs. The difference between the overall average of the two amounts to $0.46 - 0.17 = 0.29$. While this still indicates a clear distinguishability between the two, it also suggests that increasing the size of the dataset decreases the model’s performance. It can be concluded that the flexibility of a single BasicEntanglerLayer is not sufficient to reflect structures within even larger datasets. Table 6.5 shows the results for the best run of this iteration as an example. It shows that the words occurring in the same sentence are on average, always more similar than those that do not occur within the same sentence. However, the absolute amount of these similarities varies greatly. For example, while the similarity of the words ‘dogs’ and ‘cuddling’ amounts to 0.99, the similarity of ‘dogs’ and ‘bark’ is only 0.25. This also indicates that the model’s flexibility is too limited to represent structures within larger text corpora. Another cause of these inconsistencies might be the inadequacy of the trace inner product as a measure of similarity.

run	ii similarity	ij similarity
1	0.59	0.11
2	0.49	0.20
3	0.33	0.24
4	0.39	0.25
5	0.34	0.15
6	0.43	0.15
7	0.60	0.11
8	0.45	0.19
9	0.56	0.13
10	0.38	0.17
overall average	0.46	0.17

Table 6.4: Average word similarity of density matrix word embeddings on the Animal dataset with a single BasicEntanglerLayer

word 1	word 2	similarity	word 1	word 2	similarity
dogs	cuddling	0.99	dogs	independent	0.00
cats	cuddling	0.04	cats	independent	0.91
lions	cuddling	0.10	lions	independent	0.10
dogs	depend	0.87	dogs	meow	0.25
cats	depend	0.13	cats	meow	0.59
lions	depend	0.14	lions	meow	0.23
dogs	bark	0.25	dogs	wild	0.00
cats	bark	0.15	cats	wild	0.16
lions	bark	0.08	lions	wild	0.42
dogs	alone	0.25	dogs	hunt	0.25
cats	alone	0.36	cats	hunt	0.16
lions	alone	0.23	lions	hunt	0.80

Table 6.5: Detailed word similarities of density matrix embeddings of the best run on the Animal dataset with a single BasicEntanglerLayer

7

Summary and Further Work

7.1 SUMMARY

The DisCoCat model formalizes linguistic structures using mathematical concepts of quantum mechanics. It operates in high-dimensional tensor product spaces and thus requires a high amount of computational resources. However, the shared mathematical structure allows to find quantum-mechanical counterparts for word meaning vectors and grammatical structure and perform the computations on quantum hardware. Furthermore, the DisCoCat model suggests using density matrices as word embeddings. Their ability to capture probability distributions over different states proves helpful in modeling natural language phenomena like word ambiguity and lexical entailment.

Enforcing properties of density matrices in a classical neural network architecture has brought with it the need to learn intermediary matrices. These cause sub-optimal parameter optimization.

This thesis tackles this problem by proposing a hybrid quantum-classical algorithm for learning density matrix word embeddings. Preparing a variational quantum circuit into a state corresponding to a word's meaning and extracting its state yields valid density matrices without the need for intermediary matrices.

To learn $N \times N$ -dimensional density matrix word embeddings, $\log_2 N + M$ qubits are needed. Tracing the additional M qubits out allows preparing a mixed state on $\log_2 N$ qubits. The first implementation is executed on a quantum simulator to test the concept of this approach. A basic circuit ansatz with an X-rotation on every qubit and a ring of CNOT gates is implemented on five qubits. Tracing out three of them and extracting the state of the remaining two yields 4×4 -dimensional density matrices. Only five parameters were needed, while the classical Word2DM model needs 16 parameters for the same density matrices. The similarities of the learned density matrices demonstrate the model's ability to capture linguistic structure within very small text corpora. Co-occurring words are more similar than words that do not occur together. However, the results also show the limitations of the chosen circuit ansatz. With increasing vocabulary size, the model's performance decreases. Another shortcoming of this approach is the sub-optimal choice of the trace inner product

as a similarity measure.

7.2 FURTHER WORK

This thesis has laid the foundation for learning word embedding density matrices on quantum computers. While the obtained results are promising, there is still much to be explored.

7.2.1 Circuit Design

First of all, the choice of the circuit ansatz must be optimized and theoretically substantiated. The optimal number M of additional qubits to be traced out (see section 5.1) should be explored and different entangling patterns should be tested. Furthermore, the optimal trade-off between flexibility and complexity that was shortly discussed in section 5.2 should be investigated.

Finding a highly suitable circuit ansatz for the given task should allow to process larger and more complex text corpora.

7.2.2 Similarity Measure and Objective Function

The used objective function from equation 4.19 is based on the trace inner product as a similarity measure. Thus, this measure also has to be used in the evaluation. Section 4.3.1 presented the shortcomings of this choice and proposed different candidates with clear advantages for a similarity measure. Designing new objective functions based on more suitable similarity measures like the fidelity might improve the performance not only of this but also of the Word2DM model and other architectures proposed in [13].

The learned density matrices should be evaluated on word similarity datasets like [36, 37, 38, 39, 40]. Comparing the results to state-of-the-art architectures should provide detailed information on the model's quality.

7.2.3 Word Ambiguity and Lexical Entailment

Furthermore, it would be interesting to evaluate the learned density matrices in word ambiguity and lexical entailment tasks. Therefore, also the composition properties of the learned density matrices should be explored for different composition methods like those introduced in section 4.6. Comparing the results to those in [13] should provide further information about the model's quality.

Since the used objective function optimizes word similarities, designing better suitable objective functions for word ambiguity and lexical entailment based on measures like the

Von Neumann Entropy (see section 4.5) or the representativeness (see section 4.4) could improve the model's performance.

7.2.4 Implementation on Real Quantum Hardware

This thesis demonstrated the feasibility of learning density matrices with a quantum-hybrid algorithm using a quantum simulator. It would be interesting to further investigate the realization of this implementation on real quantum hardware. Therefore, a method for estimating the gradients of circuits of the form of equation 5.1 on real quantum hardware is needed.

In addition, the extraction of the density matrices at the end of the circuit must be reconsidered. Since performing quantum state tomography in each training step is computationally costly, the evaluation of components of the objective function on quantum hardware should be explored. If the trace of the product of two density matrices can be evaluated on a quantum computer, the density matrices would not have to be extracted in each training step, but only once at the end of the training process for further use. The impact of this on the gradient computation would also have to be investigated.

Regardless of whether a reduction in computational resources can be achieved by an implementation on quantum hardware, it would be interesting to compare the learned density matrices' ability to reflect linguistic structures to classically learned ones.

List of Figures

2.1	Illustration of the Bloch sphere representation of a single-qubit state ψ pointing to the sphere's surface. The angle ϕ describes the angle between the state's vector and the X-axis. Consequently, θ describes the angle between the state and the Z-axis. In addition to their designation, the respective basis states are also marked on the axes.	8
2.2	Illustration of a mixed state in the Bloch sphere. Mixed states are located inside the Bloch sphere and not on the surface. The closer a state is to the sphere's origin, the more mixed it is.	13
2.3	Illustration of the impact of the RX, RY, and RZ gates on a state in the Bloch sphere.	15
2.4	Scheme of a variational quantum circuit. The input data is encoded using the parameters ϕ , and the parameters θ are the trainable model parameters. The model's output is the expectation value $f(\phi, \theta)$ approximated by running the circuit many times and averaging the measurement results. The gradients are approximated numerically using the parameter shift rule in equation 2.19 and the parameters are updated using classical optimization techniques. . .	16
3.1	Graphical Calculus for monoidal Categories adapted from [5].	22
3.2	Graphical Calculus for Compact Closed Categories, adapted from [5].	24
3.3	Quantum circuit for the sentence 'John likes Mary'. A sequence of parameterized rotation gates prepares the single-qubit states of the nouns and the three-qubit state of the transitive verb is prepared by two IQP-layers consisting of a row of Hadamard gates followed by a chain of controlled Z rotations each. At the end of the circuit, the Bell measurements corresponding to the cups are applied.	32
4.1	Procedure for finding target-context word pairs with window size two. Context words are selected within a window size 2 before and after the target word.	54

4.2	Skip-gram architecture. The input is the target word's one-hot vector. The network has one linear hidden layer and a linear output layer with a softmax activation function. The output is a probability distribution over all words in the vocabulary of being the context word corresponding to the given target word. The loss between the output and the true context word's one-hot vector is computed to update the network's weights. The rows of the hidden layer's weight matrix provide the word embedding vectors after training. . . .	55
5.1	Architecture for learning quantum density matrix word embeddings. A parameterized unitary is applied to all qubits. Afterwards a sub-system is traced out in order to put the other qubits into a mixed state. The remaining qubits' state is extracted at the end of the circuit and the parameters are optimized according to equation 4.19.	59
5.2	A single layer of the BasicEntanglerLayer ansatz. Parameterized X-rotation gates are applied to all qubits, followed by a ring of CNOT gates.	63
5.3	Full circuit diagram of the implemented model with a single BasicEntanglerLayer. First, the operations of BasicEntanglerLayer are applied to all five qubits. Afterwards three qubits are traced out, and the density matrix representation of the other two qubits' state is extracted at the end of the circuit.	63

List of Tables

2.1	Overview of standard gates. The gate's name is on the left, the circuit diagram is located in the middle, and the corresponding operators can be found on the right.	14
3.1	This table presents the reduction of necessary qubits when using quantum hardware compared to bits on classical hardware.	33
4.1	Overview of the graphical notations of the components of CPM(FHilb) with their pendant in FHilb . Doubled wires in FHilb are simplified to single thick wires in CPM(FHilb)	39
6.1	Average word similarity of density matrix word embeddings on the ABC dataset with a single BasicEntanglerLayer	67
6.2	Detailed word similarities of density matrix embeddings of the best run on the ABC dataset with a single BasicEntanglerLayer	68
6.3	Average word similarity of density matrix word embeddings on the ABC dataset with two BasicEntanglerLayers	69
6.4	Average word similarity of density matrix word embeddings on the Animal dataset with a single BasicEntanglerLayer	70
6.5	Detailed word similarities of density matrix embeddings of the best run on the Animal dataset with a single BasicEntanglerLayer	70

Bibliography

- [1] Schütze, H.: Automatic word sense discrimination. *Computational Linguistics* 24(1), 97–123 (1998)
- [2] Mikolov, T., Chen, K., Corrado, G.s., Dean, J.: Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR* (2013)
- [3] Lambek, J.: Type grammar revisited. In: *International Conference on Logical Aspects of Computational Linguistics*, pp. 1–27 (1997). doi:10.1007/3-540-48975-4_1
- [4] Clark, S., Pulman, S.G.: Combining symbolic and distributional models of meaning. In: *AAAI Spring Symposium: Quantum Interaction* (2007)
- [5] Coecke, B., Sadrzadeh, M., Clark, S.: Mathematical foundations for a compositional distributional model of meaning. *CoRR arXiv:1003.4394* (2010)
- [6] Zeng, W., Coecke, B.: Quantum algorithms for compositional natural language processing. *Electronic Proceedings in Theoretical Computer Science* 221, 67–75 (2016). doi:10.4204/eptcs.221.8
- [7] Clark, S., Coecke, B., Grefenstette, E., Pulman, S., Sadrzadeh, M.: A quantum teleportation inspired algorithm produces sentence meaning from word meaning and grammatical structure. *CoRR arXiv:1305.0556* (2013)
- [8] Abramsky, S., Coecke, B.: A categorical semantics of quantum protocols. In: *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004.*, pp. 415–425 (2004). doi:10.1109/LICS.2004.1319636
- [9] Coecke, B., de Felice, G., Meichanetzidis, K., Toumi, A.: Foundations for near-term quantum natural language processing. *arXiv:2012.03755* (2020)
- [10] Meichanetzidis, K., Toumi, A., de Felice, G., Coecke, B.: Grammar-aware question-answering on quantum computers. *arXiv:2012.03756* (2020)
- [11] Piedeleu, R., Kartsaklis, D., Coecke, B., Sadrzadeh, M.: Open system categorical quantum semantics in natural language processing. *CoRR arxiv:1502.00831* (2015)

- [12] Balkir, E., Sadrzadeh, M., Coecke, B.: Distributional sentence entailment using density matrices. In: International Conference on Topics in Theoretical Computer Science, pp. 1–22 (2015). doi:10.1007/978-3-319-28678-5_1
- [13] Meyer, F., Lewis, M.: Modelling lexical ambiguity with density matrices. In: Proceedings of the 24th Conference on Computational Natural Language Learning, pp. 276–290. Association for Computational Linguistics, Online (2020). doi:10.18653/v1/2020.conll-1.21
- [14] Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press, USA (2010). doi:10.1017/CBO9780511976667
- [15] Wittek, P.: Quantum Machine Learning: What Quantum Computing Means to Data Mining. Elsevier Science, Amsterdam (2016). doi:10.1016/C2013-0-19170-2
- [16] Andersson, S., Asfaw, A., Corcoles, A., Bello, L., Ben-Haim, Y., et. al.: Learn Quantum Computation Using Qiskit, (2020). <http://community.qiskit.org/textbook>
- [17] Schuld, M., Petruccione, F.: Supervised Learning with Quantum Computers, 1st edn. Springer, Cham (2018). doi:10.1007/978-3-319-96424-9
- [18] Schuld, M., Bergholm, V., Gogolin, C., Izaac, J., Killoran, N.: Evaluating analytic gradients on quantum hardware. Physical Review A 99(3) (2019). doi:10.1103/physreva.99.032331
- [19] Leinster, T.: Basic Category Theory. Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge (2014). doi:10.1017/CBO9781107360068
- [20] Bankova, D., Coecke, B., Lewis, M., Marsden, D.: Graded hyponymy for compositional distributional semantics. Journal of Language Modelling 6, 225 (2019). doi:10.15398/jlm.v6i2.230
- [21] Coecke, B., Kissinger, A.: Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning. Cambridge University Press, Cambridge (2017). doi:10.1017/9781316219317
- [22] Giovannetti, V., Lloyd, S., Maccone, L.: Quantum random access memory. Physical Review Letters 100(16) (2008). doi:10.1103/physrevlett.100.160501
- [23] Heunen, C., Vicary, J.: Introduction to categorical quantum mechanics. Clarendon Press, Oxford (2014)

- [24] Esma, B.: Using density matrices in a compositional distributional model of meaning. Master's thesis, University of Oxford (2014)
- [25] Kartsaklis, D., Sadrzadeh, M.: A study of entanglement in a categorical framework of natural language. *Electronic Proceedings in Theoretical Computer Science* 172 (2014). doi:10.4204/EPTCS.172.17
- [26] Birkhoff, G., Von Neumann, J.: The logic of quantum mechanics. *Annals of mathematics*, 823–843 (1936)
- [27] Coecke, B., Meichanetzidis, K.: Meaning updating of density matrices. *FLAP* 7, 745–770 (2020)
- [28] Widdows, D., Peters, S.: Word vectors and quantum logic: Experiments with negation and disjunction. *Mathematics of Language* 8, 141–154 (2003)
- [29] Widdows, D.: Orthogonal negation in vector spaces for modelling word-meanings and document retrieval. In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, pp. 136–143. Association for Computational Linguistics, USA (2003). doi:10.3115/1075096.1075114
- [30] Coecke, B.: The mathematics of text structure. In: *Joachim Lambek: The Interplay of Mathematics, Logic, and Linguistics*, pp. 181–217. Springer, Cham (2021). doi:10.1007/978-3-030-66545-6_6
- [31] Lewis, M.: Compositional hyponymy with positive operators. In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, pp. 638–647. INCOMA Ltd., Varna, Bulgaria (2019). doi:10.26615/978-954-452-056-4_075
- [32] Leifer, M.: Quantum graphical models and belief propagation. *Annals of Physics* 323(8) (2008). doi:10.1016/j.aop.2007.10.001
- [33] Leifer, M., Spekkens, R.: Formulating quantum theory as a causally neutral theory of bayesian inference. *Physical Review A* 88 (2011). doi:10.1103/PhysRevA.88.052130
- [34] Coecke, B., Spekkens, R.: Picturing classical and quantum bayesian inference. *Synthese* 186 (2011). doi:10.1007/s11229-011-9917-5
- [35] McCormick, C.: Word2Vec Tutorial - The Skip-Gram Model. (2016). <http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/>

- [36] Rubenstein, H., Goodenough, J.B.: Contextual correlates of synonymy. *Communications of the ACM* 8(10), 627–633 (1965). doi:10.1145/365628.365657
- [37] Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., Ruppin, E.: Placing search in context: The concept revisited. *ACM Trans. Inf. Syst.* 20(1), 116–131 (2002). doi:10.1145/503104.503110
- [38] Miller, G.A., Charles, W.G.: Contextual correlates of semantic similarity. *Language and Cognitive Processes* 6(1), 1–28 (1991). doi:10.1080/01690969108406936
- [39] Hill, F., Reichart, R., Korhonen, A.: Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics* 41(4), 665–695 (2015). doi:10.1162/COLI_a_00237
- [40] Bruni, E., Boleda, G., Baroni, M., Tran, N.-K.: Distributional semantics in technicolor. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, vol. 1, pp. 136–145 (2012)

Statement of Originality

I hereby confirm, that I have written the accompanying thesis by myself, without contributions from any sources other than those cited in the text and acknowledgements. This work has not yet been submitted to any examination office in the same or similar form.

Osnabrück, April 26, 2022

Saskia Bruhn

