Master's Thesis

in Computational Linguistics

at the Ludwig-Maximilians-University Munich

Faculty 13

Analyzing word predictions by quantum natural language processing

submitted by
Jakob Murauer

Supervisors:	M.Sc. Korbinian Staudacher and Dr. Wolfgang Gehrke
Examiner:	Prof. Dr. Michaela Geierhos
Work period:	13.03.2023 - 31.07.2023

Acknowledgments

I would like to express my sincere gratitude to all those who supported and guided me throughout the completion of my master's thesis. This research would not have been possible without the support and valuable contributions of numerous individuals.
First and foremost, I want to sincerely thank both of my supervisors Korbinian Staudacher and Wolfgang Gehrke for their continuous support throughout the thesis. I am incredibly grateful for your numerous helpful insights and long meetings, which helped me to achieve my goal. I am also very grateful for the supervision from Benjamin Rodatz. Thank you very much for your important help with the DisCoCat framework.
I am also expressing huge thankfulness to my parents, especially to my father. Without their support, my academic journey would not be possible. I also want to thank the Universität der Bundeswehr München for the access to the physical backends in the frame of the IBM Q-network and the Leibniz-Rechenzentrum for providing me access to their cloud computing service. Last but not least, I thank Prof. Dr. Geierhos for giving me the opportunity to work in this exciting field of quantum computing and for supervising my work.

Abstract

English version:
Quantum natural language processing is an emerging field, that combines principles of quantum computing with natural language processing with the aim to enhance the capabilities to process and analyze human language. In classical natural language processing, word prediction is an ingredient of the pretraining task for large language models. This forms the motivation behind the exploration of a word prediction task in QNLP.
This thesis deals with the implementation of a word prediction task in quantum natural language processing using a mathematical framework, particularly the DisCoCat framework. Firstly, we reformulate word prediction as a binary classification task and train quantum machine learning models on that task. Secondly, we implement a word prediction task as a multiclass classification and also train models on that. In the course of this implementation, we give a comprehensive explanation of how to design a multiclass classification in the DisCoCat framework. Afterwards, we show how masking can be conceptualized inside a quantum computing environment. Then, a strategy is presented, which reduces the number of qubits and lowers the task complexity. This strategy has the potential to be extended beyond word prediction and to be applied to various other problems as well.
The evaluation of the models which have been trained with the binary classification task shows promising results. The masking approach and the strategy for reducing the number of qubits were both a success in the evaluation. However, the evaluation of the models trained on the multiclass classification approach is not of the same standard as the binary approach. The reasons for that outcome are discussed in depth. For both tasks, we document the applied hyperparameters. Further research topics include the continued development of multiclass classification in the DisCoCat framework and research on new ansätze in quantum machine learning.

German version:

Quantum natural language processing ist ein aufstrebendes Gebiet, das die Prinzipien von Quantencomputing mit den Methoden der Computerlinguistik verbindet, mit dem Ziel, die Verarbeitung und Analyse von menschlicher Sprache zu verbessern. In der klassischen Sprachverarbeitung kommt das Training anhand von Wortvorhersageaufgaben bei großen Sprachmodellen wiederholt vor. Dies formt die Motivation für die Erforschung einer Wortvorhersageaufgabe in QNLP.

Diese Arbeit beschäftigt sich nun mit einer Implementierung dieser Aufgabe in QNLP mithilfe des mathematischen Frameworks DisCoCat. Zuerst wird die Wortvorhersageaufgabe als binäre Klassifikationsaufgabe neu formuliert. Wir trainieren mehrere quantum machine learning Modelle anhand dieser binären Aufgabe. Später implementieren wir die Wortvorhersageaufgabe auch als mehrklassige Aufgabe. Auch hier werden mehrere Modelle trainiert. Im Zuge dessen geben wir eine detaillierte Erklärung, wie eine mehrklassige Aufgabe im DisCoCat Framework erstellt werden kann. Außerdem zeigen wir, wie eine Maskierung eines Wortes in einer Quantencomputingumgebung realisiert werden kann. Danach präsentieren wir noch eine Strategie, welche sowohl die Anzahl der Qubits verringern kann, als auch die Schwierigkeit der Aufgabe senken kann. Diese Strategie hat das Potential auch auf andere Aufgabengebiete angewendet werden zu können fern der Wortvorhersagung.

Die Evaluation der Modelle, welcher mit der binären Klassifikationsaufgabe trainiert worden sind, ist vielversprechend. Der Maskierungsansatz und die Strategie zur Verringerung von Qubits zeigen auch sehr positive Ergebnisse. Die Evaluation der Modelle mit dem mehrklassigen Ansatz fällt schlechter aus als die Evaluation der binär trainierten Modelle. Die möglichen Gründe dafür werden ausgiebig besprochen. Zudem werden alle verwendeten Hyperparameter detailliert dokumentiert. Als weitere Forschungsgebiete werden unter anderem mehr Forschung für mehrklassige Aufgaben im DisCoCat Framework und die weitere Entwicklung von variationellen Schaltkreisen in Quantumcomputing genannt.

Contents

1. Introduction

In recent years, quantum computing has made remarkable advancements in the field of computer science. This field has garnered significant attention due to its theoretical ability to solve tasks that are computationally infeasible using classical methods within a reasonable timeframe. With the advent of quantum computing, the research area of quantum machine learning emerged. This subfield has the goal to speed up the training and evaluation time of machine learning models, while also reducing the number of parameters. Essentially quantum machine learning has the goal to optimize machine learning with the help of quantum computing.
Parallel to that development, the field of categorical quantum mechanics emerged. Here, quantum mechanics is described through category theory, an innately diagrammatic branch of mathematics. By discovering that a grammar in linguistics and the Hilbert space (which is the mathematical space in which quantum computing is positioned) both form the same category in category theory, the DisCoCat framework was formed. In this framework, syntactical grammar and semantics are merged with the help of categorical concepts. The DisCoCat framework can be natively combined with quantum computing, where information of words and sentences is processed with support from quantum mechanics.
Meanwhile in natural language processing, the advent of large language models in classical natural language processing has made huge progress in the capability of processing and analyzing human language. Training large language models comes with the question of how can we train the models, in order to let them learn human language. In the prominent language model BERT, masked language modeling is incorporated. This task refers to the prediction of a missing word in a sequence of words.

Quantum natural language processing is a research area that combines all of these three fields with the purpose of enhancing natural language processing. Prior work in the field of quantum natural language processing focused on binary sentence classification and sentiment analysis and not on the exploration of pretraining approaches in QNLP. The

research question of this thesis therefore is:

How can a word prediction task be implemented in QNLP, and how does this task perform using a quantum machine learning setup?

The motivation for this research is the possibility that quantum technology could be used to improve language models. Therefore, this thesis explores word prediction in quantum natural language processing at a fundamental level. The resulting problem sets were: How can a masking strategy be developed in terms of quantum computing? This is essential since we need to conceal the information of the word that the model needs to predict.
How can the number of qubits be kept low? This is an often occurring goal in quantum computing since we are currently very limited in the number of qubits we have on quantum hardware. It is also extremely computationally expensive to simulate qubits classically.
How can a multiclass classification task be realized in the DisCoCat framework? In the future, the DisCoCat framework is aimed to tackle more complex problems than just binary classification problems. It is important to show, how multiclass classification can be accomplished in QNLP.
What are good fitting parameters for such a task, and how much does a model need to be scaled? With the methods we will present in this thesis, it is reasonable to show the hyperparameters found, so that future works can be built upon this thesis.

1.1. Outline

This thesis will first describe masked language modeling. Masked language modeling is a technique used for training language models, where the model has to predict a masked word within a sequence of words. It is found in numerous successful language models. This explanation is helpful since an approach to masked language modeling is later implemented into a quantum natural language processing framework.
Chapter 3 deals with a thorough introduction to the field of quantum computing. This chapter shows the fundamental laws of quantum computing and guides the reader through the field of quantum machine learning. This is extensively needed since the used DisCoCat framework is later combined with a quantum machine learning setup in the experimental part of this thesis.

In chapter 4, the reader is provided with a comprehensive introduction to category theory. Firstly, the thesis explains the basic notions of this branch of mathematics. After that, the necessary categorical concepts for the DisCoCat framework are explained. This is important since the DisCoCat framework is inherently categorical.
The following chapter 5 explains the DisCoCat framework, which will be used in the experimental part. In this chapter quantum computing and category theory are combined, in order to explain quantum natural language processing. The DisCoCat framework forms the foundation for current quantum natural language processing.
In the experimental part in chapter 6 we combine every topic explained so far. We implement a pretraining approach, which is found in classical language models, in a quantum machine learning setup using the DisCoCat framework, which is a categorical concept by nature. We document the implementation of the pretraining approach and train small models on a toy dataset. We evaluate their performances and show how different hyperparameters perform in quantum natural language processing.
In Chapter 7, we conclude our findings and bring them into a more general context. Since quantum natural language processing and quantum machine learning are very recent fields, we give several further research topics.

2. Masked language modeling

This chapter will introduce the concept of language modeling in classical natural language processing (NLP). Specifically, this chapter is dedicated to introducing the reader to masked language modeling (MLM), a critical task that involves predicting a token within a sequence of words. An understanding of this concept is helpful since the experimental part of this thesis will feature an approach to implementing masked language modeling into quantum natural language processing. We will also give a very short introduction to transfer learning in order to bring MLM into a better NLP context.

2.1. Overview

Language modeling is an essential task in NLP, where the goal is to predict a token or word given a sequence of words. One approach to language modeling is masked language modeling. MLM is a widely used training technique, where the model has to predict a randomly hidden (masked) word based on the context of the surrounding words. It is designed to enable language models to acquire a comprehensive understanding of natural language. By randomly masking certain words or tokens in a sentence and prompting the model to predict these masked words based on context, MLM encourages the model to learn language structure and semantics. This learning phase is usually called pretraining. The MLM approach is widely taken by language models like BERT or RoBERTa. These models are based on neural network architecture, in particular, they are Transformer-based models [14]. For a detailed view of transformer models for natural language processing, the reader is referred to [48].
MLM is an inherently bidirectional approach since the model takes information from both the right and the left context. We will see later in this thesis, that we have bidirectionality in quantum natural language processing as well.

2.2. MLM implementation

This section deals with concrete details of the MLM implementation taken in BERT. BERT, Bidirectional Encoder Representations from Transformers, is a large neural language model based on Transformer architecture released in 2018. For a full understanding of BERT's architecture, the reader is referred to [14]. Likewise, for a thorough explanation of the Transformer architecture and the attention mechanism, we refer to [61].
For the initial pretraining phase, two tasks are used, masked language modeling and next sentence prediction. For the sake of this thesis, we will focus on MLM. In the masked language modeling process, a moderate amount of tokens is randomly sampled and for every randomly sampled token, a prediction needs to be performed by the model [14]. The following list shows the exact process:

- 15% of the tokens are randomly sampled
  - 80% of the sampled tokens are replaced by a $[MASK]$ token.
  - 10% are replaced by a random token.
  - The last 10% are left unchanged.

In the following examples, masked language modeling is described. Note that $[CLS]$ is a classification token and is used as the first input for every sequence [14]:

- $[CLS]$ the man went $[MASK]$ the store and bought milk
  The predicted token should be *to*.

- $[CLS]$ the quick brown $[MASK]$ jumps over the lazy dog
  The predicted token should be *fox*.

In [33] the authors present an optimized pretraining approach to BERT. Here, the next sentence prediction is entirely dropped, focusing solely on MLM [33]. The authors stated specifically that removing the next sentence prediction matches or moderately improves downstream task performance [33].
In the RoBERTa (Robustly Optimized BERT Pretraining Approach) report, a new masking variation is also presented. In BERT masking is done once before the training starts and the language model is presented by the same masking over all epochs. This method of masking can be called static masking [33]. In RoBERTa, dynamic masking is introduced. The masking is created in 10 different ways, so RoBERTa gets presented with

different maskings over the course of training [33]. Their improved pretraining approach includes a larger training set, longer sequences, and extended training duration too. This leads to state-of-the-art results (as of the release of the paper) on multiple evaluation metrics, reinforcing the significance of pretraining approaches for language models [33].

2.3. Transfer learning

The question arises why are language models trained on such an arbitrary task like word prediction when they are often used for very concrete tasks such as sentiment analysis, hate speech detection, machine translation, and various more? The reason for that lies in the concept of transfer learning. Transfer learning is a technique where a model is trained on one task in order to improve the performance on various other related tasks [60]. The key idea behind transfer learning is that knowledge learned from one task can be transferred to a different but related task, often leading to a general performance gain. By using masked language modeling in transfer learning, the model becomes proficient in understanding language at a deep level, and this knowledge can be effectively transferred to various downstream NLP tasks, resulting in improved performance and reduced training time compared to training each task from scratch [60]. Huge language models like BERT or RoBERTa are pretrained once extensively on MLM and afterwards finetuned on a specific task.

3. Quantum computing

Quantum computing is a new field in computer science combining aspects of quantum mechanics and computer science. It explores the potential of quantum systems completing tasks faster than classical systems. The key aspects of quantum computing will be discussed in this section given that nearly all the computational aspects of the experimental part rely on quantum computing. This chapter is mostly based on [44] and [23].

3.1. Dirac-notation and Hilbert space

The Dirac, or bra-ket-notation, is of utmost importance in the field of quantum mechanics and quantum computing. Let v be a vector of an n-dimensional complex vector space. $|v\rangle$ (ket) denotes a column vector and the counterpart $\langle v|$ (bra) denotes a row vector:

$$|v\rangle = \begin{pmatrix} v_1 \\ v_2 \\ ... \\ v_n \end{pmatrix} \qquad \langle v| = (v_1^*, v_2^*, ..., v_n^*)$$

Note that the bra notation is not just the transposed ket vector, but the row vector of ket with the complex conjugated values. The notation $\langle\psi|\phi\rangle$ denotes the scalar product. Hilbert Spaces are structures constructed upon vector spaces, in particular, complex vector spaces, that are combined with an inner product [21]. Hilbert spaces are extensively used in quantum information theory, as they enable the computation of angles and distances between vectors and therefore quantum states. This phenomenon is used for calculating measurement probabilities in quantum theory [21].

Definition 3.1.1. An inner product on a complex vector space V is a function $\langle x|y\rangle : V \times V \to \mathbb{C}$ with the following properties [21]:

- It is conjugate-symmetric: For all $a, b \in V$ it holds: $\langle a|b\rangle = \langle b|a\rangle^*$
- It is linear in the second argument: For all $a, b, c \in V$ and $s \in \mathbb{C}$ it holds: $\langle a|s \cdot b\rangle = s \cdot \langle a|b\rangle$ and $\langle a|b + c\rangle = \langle a|b\rangle + \langle a|c\rangle$
- It is positive definite: For all $a \in V$ it holds: $\langle a|a\rangle = \sum_{i=1}^{n} a_i^* a_i$
  Thus: $\langle a|a\rangle \in \mathbb{R}$ and $\langle a|a\rangle \geq 0$ and $\langle a|a\rangle = 0 \Rightarrow a = 0$

Definition 3.1.2. In a complex vector space with an inner product V one norm of an element v can be defined as $||v|| = \sqrt{\langle v|v\rangle}$ [21]. Whereas the inner product of two complex numbers is $\langle a|b\rangle = \sum_{i=1}^{n} a_i^* b_i$ [21].

It is possible to view the bra-ket multiplication in terms of matrix-multiplication since a bra vector can be thought of as a matrix of form $1 \times n$ and the corresponding ket as a matrix of form $n \times 1$. The multiplication of bra and ket yields a matrix of form 1×1, which is isomorphic to a scalar.
Hilbert spaces are not mentioned without concrete reason here. Firstly, every quantum state in quantum computing lives inside a finite-dimensional Hilbert space [17]. Secondly, a finite-dimensional Hilbert space has important properties from the point of view of category theory. It can facilitate mappings, which in turn enable natural language processing with the help of quantum computing. These concepts will be explained in chapters 4 and 5. For a more detailed description of the mathematics behind Hilbert spaces the reader is referred to [49] and [17].

3.2. Qubits

The basis of classical computing is the manipulation of bits, which can be in either two states, 0 and 1. The basis of quantum computation revolves around the manipulation of quantum bits, called qubits.
A qubit is mathematically described as:

$$\alpha \cdot |0\rangle + \beta \cdot |1\rangle \, (\alpha, \beta \in \mathbb{C})$$

$|0\rangle$ and $|1\rangle$ are defined as $\binom{1}{0}$ and $\binom{0}{1}$ respectively. Mathematically speaking, a qubit is a linear combination of two basis states.

α and β, which are both complex numbers, are the probability amplitudes of the qubit. $|\alpha|^2$ describes the probability of measuring the state $|0\rangle$ and $|\beta|^2$ describes the probability of measuring $|1\rangle$. For every qubit it holds:

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

With this information, one can see the major difference between quantum computing and classical computing. A quantum bit is not only 0 or 1 but can have probabilities of both states simultaneously. Consider the following state:

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

The aforementioned statement, that both probabilities add up to 1, holds because both probabilities $|\alpha|^2$ and $|\beta|^2$ are 0.5. Any state, in which a qubit can exist in multiple states simultaneously, is called a superposition. A more detailed mathematical description will follow in section 3.3.1.

Through measurement, one cannot simply observe α or β. Measurement reveals either $|0\rangle$ or $|1\rangle$. When measuring a qubit in superposition, the superposition of the example above gets destroyed and the qubit collapses with a probability of 0.5 to either $|0\rangle$ or $|1\rangle$. When measuring $|\phi\rangle$ 1000 times, one should expect to get $|0\rangle$ and $|1\rangle$ around 500 times.

The Bloch sphere is a geometric representation used to visualize the state of a single qubit. It provides an intuitive way to understand the quantum state of a qubit as a point on the surface of a sphere [23]. The Bloch sphere has three axes, X, Y, Z. The following figure shows a plain representation of the Bloch sphere on the left and a Bloch sphere with two states depicted on it on the right. The states on the right are color-coded, such that the blue arrow represents the state $|1\rangle$ and the red arrow represents $|0\rangle$.

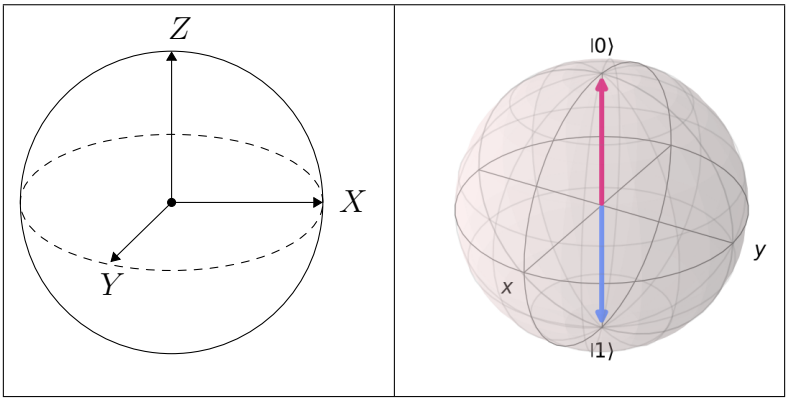


Figure 3.1.: Representation of the Bloch sphere

3.3. Quantum gates

The core of computing is the manipulation of states. Just like in classical computing with normal bits, qubits get manipulated with gates. The mathematical representation of a gate is a unitary matrix. A unitary matrix is an invertible complex square matrix of which the product of itself with its conjugate transpose is the identity matrix [24]. In other words, a complex square matrix is unitary, if its conjugate transpose is its inverse. Let U denote the unitary matrix, the operator $*$ the conjugate, T the transpose, and I the identity matrix.

$$U^{*T}U = U^{\dagger}U = U^{-1}U = I$$

The conjugate transpose is also called the Hermitian adjoint and is denoted by a dagger †. Unitary matrices play a big role in quantum computing, because of their norm-preserving property. That means when multiplying a unitary matrix, e.g. a gate, with a vector, e.g. a qubit, the property that the probability amplitudes of our qubit represent a probability distribution gets preserved, and thus the probabilities still add up to 1. Visually speaking, when multiplying a gate with a qubit the direction of the vector gets changed, while the length of the vector gets preserved. Thus, the qubit is still on the aforementioned Bloch sphere after a gate manipulation.
The second important property of unitary gates is reversibility. Every manipulation in quantum computing needs to be reversible, as seen above.

3.3.1. Single qubit gates

Single qubit gates are used to bring a qubit in one state to a new state. Important single qubit gates are Pauli-gates (X,Y,Z gates), the Hadamard gate, and for quantum machine learning, arbitrary rotation gates. The following table gives a short overview of the standard gates.

	X	**Y**	**Z**	**H**
Matrix	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
Rotation	π around X	π around Y	π around Z	$\frac{\pi}{2}$ around Y followed by π around X

Table 3.1.: Common quantum computing gates

Rotation gates also represent a rotation around one of the three axes but with an arbitrary rotation angle θ. These three rotational gates are of the following form:

$$R_X = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \qquad R_Y = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \qquad R_Z = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$$

The Hadamard gate, denoted as H in the table, is arguably one of the most important tools in quantum computing. It is a single-qubit gate that maps the basis states $|0\rangle$ or $|1\rangle$ to a superposition of equal probabilities of both basis states. In superposition, a quantum state is in multiple states simultaneously. This is a powerful concept that is used in various renowned quantum algorithms such as Grover's algorithm [18] and the quantum Fourier transform [7].

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$H|1\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

The resulting states in the equations above are called superpositions. They can also be written as $|+\rangle$ and $|-\rangle$ respectively.
Any unitary single qubit operation can be decomposed into ZYZ rotation with appropriate $\alpha, \beta, \gamma, \delta$ [44]. Consider the following unitary matrix, which can represent any rotation, including a global phase:

$$U = \begin{bmatrix} e^{i(\alpha-\frac{\beta}{2}-\frac{\delta}{2})}cos\frac{\gamma}{2} & -e^{i(\alpha-\frac{\beta}{2}+\frac{\delta}{2})}sin\frac{\gamma}{2} \\ e^{i(\alpha+\frac{\beta}{2}-\frac{\delta}{2})}sin\frac{\gamma}{2} & e^{i(\alpha+\frac{\beta}{2}+\frac{\delta}{2})}cos\frac{\gamma}{2} \end{bmatrix}$$

This unitary U follows from the algebraic rules of the following formula:

$$U = e^{i\alpha}R_z(\beta)R_y(\gamma)R_z(\delta)$$

This concept of decomposition can be extended to any two non-parallel rotation axes $\hat{m}$ and $\hat{n}$ [44]. For the proof of this statement, the reader is referred to [44] page 176.

3.3.2. Multi-qubit gates

Quantum computing usually involves the manipulation of multiple qubits simultaneously. States that involve more than one qubit are represented by the tensor product $\otimes$. A quantum state $|\phi\rangle$, which has two qubits in the 0 state is therefore represented as:

$$|\phi\rangle = |0\rangle \otimes |0\rangle = |00\rangle$$

In quantum computing, there are quantum gates, that act on more than one qubit. An example of that is the $CNOT$ gate. The $CNOT$ gate is a controlled X gate. That means that the state of one qubit (called the control bit) determines the activation of X gate on another qubit (called the target bit). The following figure shows the matrix representations of the CNOT gate and the application of this gate on computational basis states:

Input state	Output state
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|01\rangle$
$\|10\rangle$	$\|11\rangle$
$\|11\rangle$	$\|10\rangle$

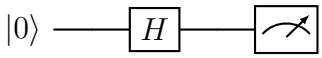
$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

The idea of a control qubit controlling the activation of a gate on another qubit can be extended to other gates as well. The reader will see in section 3.5.2 that a control operation is added on a R_y gate for example. The number of control qubits can be extended too. One can construct a multi CNOT gate, where there are multiple control bits determining the activation of an X gate on a target qubit.

3.4. Quantum circuits

Quantum circuits are a fundamental part of quantum computing. Quantum circuits are used to process information encoded in one or more qubits. Usually, quantum circuits are depicted graphically. In a quantum circuit diagram, qubits are drawn as wires, and gates are represented as rectangular boxes. The type of gate is described by the capital letter inside it. The CNOT gates make an exception, as depicted in section 3.4.1. The control qubit is simply drawn as a black dot, which is connected through a vertical wire to a target. The target qubit is portrayed as a cross with a ring around it.
The flow of information in the circuit is from left to right, where on the left end the starting state is depicted, which is usually $|0\rangle$. When obtaining the result of the quantum circuit, one or more measurements are performed.

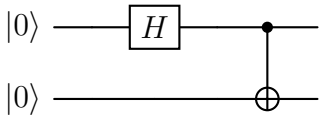
One of the simplest quantum circuit is a random number generator of the following form:



As we follow the flow of information from left to right, the $|0\rangle$ initialized qubit gets transformed into the aforementioned $|+\rangle$ state. Afterwards, a measurement is performed on the computational basis. We observe $|0\rangle$ or $|1\rangle$ at exactly 0.5 probability. The key operation at this circuit is the Hadamard gate, which puts the qubit in superposition with equal probability. This circuit can be extended with any amount of qubits, to generate a bigger range of random numbers.

3.4.1. Entanglement

Besides superposition, there exists a second important and widely used phenomenon in quantum computing: entanglement. Consider the following simple 2-qubit circuit:



The output state $|\phi\rangle$ is described as:

$$\begin{aligned} |\phi\rangle &= CNOT * (H \otimes I) * |00\rangle \\ &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \end{aligned}$$

The interesting part with this circuit is, that the output states of both qubits are always equal. That means, if we measure one qubit, we immediately know the state of the other qubit, without looking at it. This mechanism works instantly, even when the qubits are spatially separated. Examples of entangled states are the four Bell states. The state described above is one of the four Bell states. A comprehensive description of these states can be found in appendix B.

3.5. Quantum machine learning

The integration of quantum computing into machine learning gives rise to new opportunities and improvements in the field of machine learning. In this section, we will give a

short overview of relevant topics in quantum machine learning in the context of quantum natural language processing and we will describe several ansatz choices that are used in our experiment.

3.5.1. Overview

In quantum machine learning, further referenced as QML, there exist four different approaches [45].

Approach	Description
CQ	Using classical data with QML algorithms
CC	Using classical data with quantum computing inspired classical algorithms
QC	Using quantum data with classical machine learning algorithms
QQ	Using quantum data with quantum machine learning algorithms

Table 3.2.: Four approaches to quantum machine learning

Quantum machine learning, specifically the CQ approach, differs from classical machine learning in several key aspects:

Data Encoding: In classical machine learning, data is typically represented as classical bit strings or numerical vectors. In QML, data encoding involves mapping classical data to quantum states. This can be done using techniques like quantum feature maps, where classical data is transformed into quantum states using quantum gates (arbitrary rotation gates) and operations.

Model Encoding: In classical machine learning, models are constructed using mathematical functions or algorithms that operate on classical data. In QML, model encoding refers to representing the machine learning model as a quantum circuit, usually a parameterized quantum circuit. These quantum circuits are designed to encode the learning task and can be initialized with random or predefined parameters. Parameterized quantum circuits will be explained in the next section.

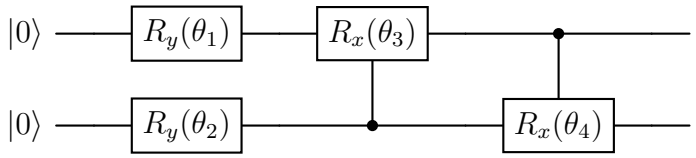
Measurements: In classical machine learning, predictions are made by evaluating the output of the model. The output is usually a vector or matrix, which is the result of a mathematical function with a certain input. In QML, measurements are performed on the final state of the quantum circuit to extract information about the output of the model. This information is probabilistic in nature, requiring statistical analysis to obtain meaningful results.

Adjustment of Parameters: In classical machine learning, model parameters are op-

timized using optimization algorithms such as Adam [30] or AdamW [35]. In QML, the adjustment of parameters is typically achieved through the use of different optimization techniques, such as SPSA. Optimization will be dealt with in section 3.5.3.
Overall, quantum machine learning, particularly the CQ approach, leverages the principles of quantum mechanics to encode data, form models, process information, and perform measurements in a fundamentally different way compared to classical machine learning.

3.5.2. Parameterized quantum circuits

Parameterized quantum circuits or variational circuits are a fundamental building block in QML [45]. This type of circuit primarily features rotation gates as their main type of gates. As described in section 3.3.1 all rotation gates have a variable rotation angle between $[0; 2\pi]$. This range is sometimes converted to $[0; 1]$ for simplification. The following circuit is a simple two-qubit variational circuit:



A parameterized circuit can be described as a unitary Operation U_θ where θ is a set of tunable parameters, acting on a quantum state $|\phi_0\rangle$ [45]. This can be mathematically described as $U_\theta |\phi_0\rangle$. Note that $|\phi_0\rangle$ is sometimes initialized as $|0\rangle^{\otimes n}$, where n is the number of qubits in the circuit [45]. The choice of the parameterized circuit is of importance, since it directly impacts the learning capability of our model [25]. In [56] the authors describe two measures, expressibility and entangling capability, for describing parameterized quantum circuits.

Expressibility

Expressibility refers to the potential of covering as many quantum states as possible on the Bloch sphere. In the following figure, two variational circuits are displayed. Both are plotted with 2000 different output states. Each output state is one blue dot on the Bloch sphere:

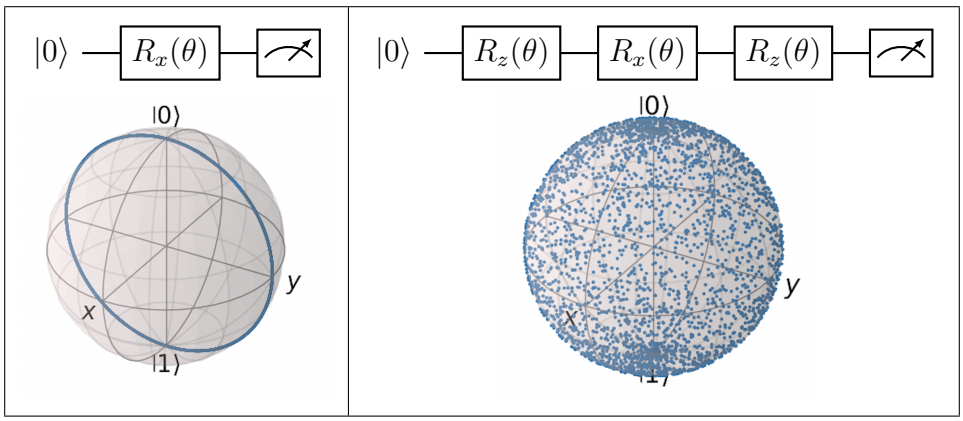
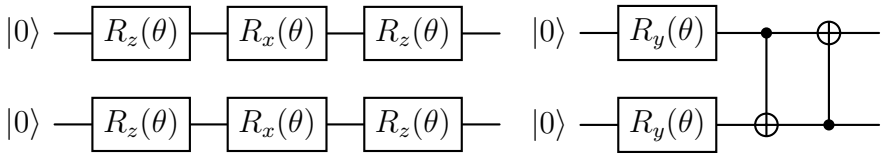


Figure 3.2.: Visualization of expressibility

It is clearly shown in this figure, that the choice of the circuit directly impacts the area covered on the Bloch sphere.

Entangling capability

Entangling capability, or Meyer-Wallach measure, is a metric that describes how entangled a given state is [56]. The measure of an unentangled product state is 0, whereas the measure for a highly entangled state, such as the Bell state, is 1. In [56] the authors define the Meyer-Wallach measure for an entire variational circuit as the average Meyer-Wallach measure for the output states it can generate. Consider the two following circuits:



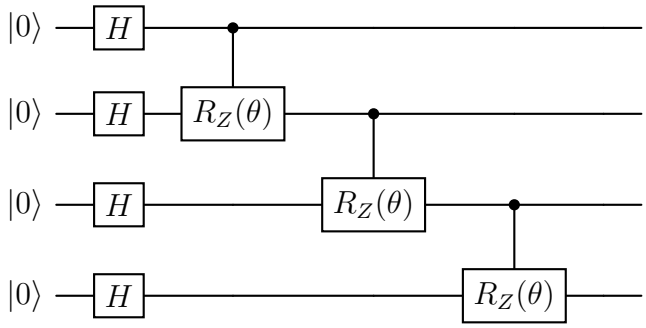
The left circuit cannot produce any entangled product state and therefore has a Meyer-Wallach measure of 0. The right circuit is able to produce entangled product states and has a Meyer-Wallach measure of greater than 0.

In [25] the authors did an empirical study on the correlation between classification accuracy on QML algorithms and expressibility and entangling capability. They found a strong correlation between classification accuracy and expressibility and a weak correlation between classification accuracy and entangling capability [25].

The next section will describe specific subroutines for choosing which gates to apply on which wires. These subroutines on variational circuits are called ansätze (singular ansatz). One block of these variational circuits is then called a layer. Each θ depicted in the subsequent circuits represents a distinct value. The ansatz versions here are the versions from the lambeq [28] package since this package is extensively used in our work. Unfortunately, different packages provide slightly different realizations of these ansätze.

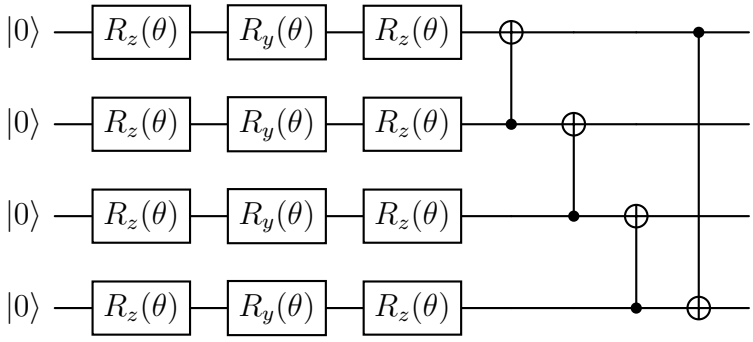
Instantaneous quantum polynomial ansatz

The instantaneous quantum polynomial ansatz, further referenced as IQP, uses a layer of Hadamard gates and diagonal unitary gates [19]. In [28] the unitary operations are implemented with controlled R_Z gates. The following 4-qubit circuit is representative of one layer of this ansatz:



Strongly entangled ansatz

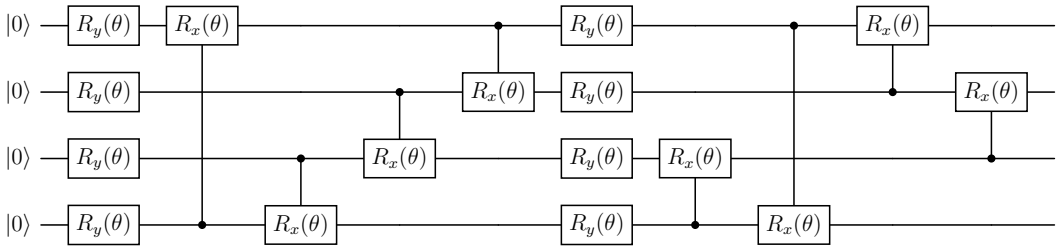
In [53] the authors propose a low-depth variational algorithm with entanglement capabilities. The figure below describes a 4-qubit example with one layer:



The implementation proposed in [28] also uses different ranges of the CNOT gates per layer, so the first layer might not match the second layer in terms of the positions of the CNOT gates.

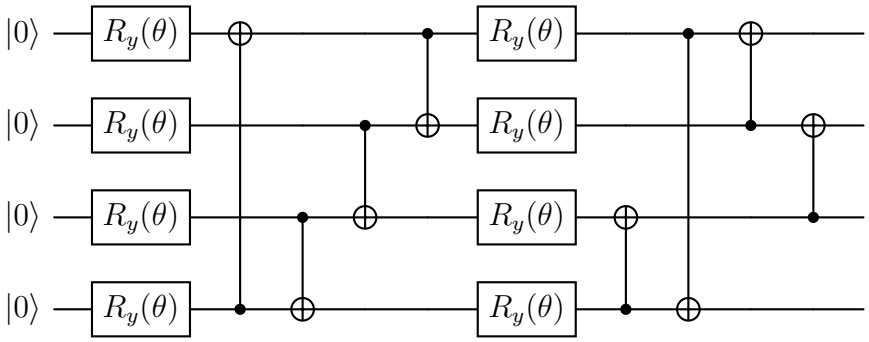
Sim 14 ansatz

Sim et al. discussed several circuits in [56]. Two of those proposed circuits have been used for our experiment in chapter 6. The circuit with number 14 in [56] is of the following form:



Sim 15 ansatz

The circuit with number 15 in [56] is of the following form:



3.5.3. Training

The learning process in machine learning models is the minimization of a cost function $f(\vec{\theta})$ with respect to the model parameters $\vec{\theta}$. A cost function quantifies how well a machine learning model performs by comparing the predicted labels to the target labels. To minimize such a function, the gradient of the cost function is calculated. With the help of gradients, specifically the negative gradients, we find the steepest decrease. This essentially reveals a direction for the desired minimum. This minimization process is called learning. The graph below is an example of a gradient ∇f at a point alongside the cost landscape.

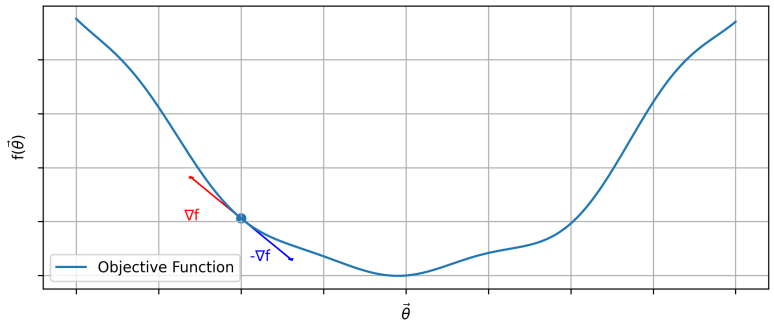


Figure 3.3.: Gradient descent

The gradient points to the direction of the steepest increase. By flipping the sign of the gradient we follow the direction of the steepest decline. We update our model accordingly:

$$\theta_{t+1} = \theta_t - \eta * \nabla f(\theta_t)$$

This method of optimization is called gradient descent. As the formula shows, gradient descent also has to deal with a stepping constant η. This stepping constant also called the learning rate, determines how big the update step at one point should be.
Calculating the gradients for models is no trivial task, especially in the context of quantum machine learning. For quantum machine learning, we need to calculate the gradients of quantum circuits [45]. A simple way of approximating gradients is by using finite differences. The core idea for approximating the gradient with this idea at point $f(\theta)$ is to get $f(\theta + \epsilon)$ and $f(\theta - \epsilon)$ and calculate the difference:

$$\nabla f(\theta_t) \simeq \frac{f(\theta + \epsilon) - f(\theta - \epsilon)}{2\epsilon}$$

The downside of using finite differences is that they only approximate gradients. Especially for noisy functions, as for example quantum circuits on near-term quantum devices, the true gradient and approximated gradient can vary strongly. The authors in [52] introduce an astonishing method for analytically, i.e. exact gradients whenever a derivative exists, calculating gradients of quantum circuits. Their method, the parameter shift rule [52], looks similar to finite differences. The following simplified parameter shift rule holds for quantum circuits with only Pauli rotations [45]. Pauli-rotations refer to a class of quantum gate operations that are based on the three Pauli matrices. Pauli-rotations are the X,Y, and Z gates. $f(\theta)$ denotes our model function with respect to the parameters θ and e_i signifies an n-dimensional zero-vector with a 1 at the position

i, where n is the number of parameters:

$$\frac{\delta f}{\delta \theta_i} = \frac{f(\theta + \frac{\pi}{2} e_i) - f(\theta - \frac{\pi}{2} e_i)}{2}$$

The downside of this approach is, that for N amount of parameters we need $2N$ calculations for the gradient since every parameter needs to be shifted downwards and upwards with a value of $\frac{\pi}{2}$. In the context of machine learning, where with large models the number of parameters increases strongly, the parameter shift rule is costly. In classical machine learning, automatic differentiation systems are being used. With automatic differentiation, gradients are highly efficiently computed using the fact that computer programs only use a handful of basic arithmetic operations. The gradients are computed by accumulating intermediate steps and applying the chain rule to them. The exact runtime can vary depending on the specific implementation and the computational requirements of the function being differentiated. However, most times the automatic differentiation program has a runtime of N with a small constant factor attached to it [37] [3]. In the context of quantum computing with noisy quantum hardware the expected $2N$ circuit evaluations for the gradient introduce additional noise, which is also something one wants to avoid. Therefore, we want to minimize the number of steps for the optimization as much as possible.

Simultaneous Perturbation Stochastic Approximation

Simultaneous Perturbation Stochastic Approximation, further referenced as SPSA, is an optimization technique to approximate the gradient [50]. The approximation technique is frequently used in quantum machine learning [34] [39]. The idea behind this approach is to randomly perturb all parameters simultaneously. One time the method perturbs downwards and one time upwards. Then it approximates the gradient via the finite differences method. Let $f(\theta)$ be the loss function, Δ a random perturbation vector with the size of the number of parameters and c a parameters shift scaling factor. The SPSA method looks as follows [16]:

$$\nabla f(\theta) \approx \frac{f(\theta + c\Delta) - f(\theta - c\Delta)}{2c} \Delta$$

The main difference is the simultaneous shift of all parameters. This is reflected in the complexity of this algorithm. As previously discussed, the standard parameter shift rule has a linear complexity $\mathcal{O}(2N)$ and the SPSA has a constant complexity $\mathcal{O}(2)$ [16].

3.5.4. Problems and limitations

A problem quantum machine learning faces is vanishing gradients that occur with an increasing number of qubits in variational circuits [38]. Vanishing gradients are associated with a flat cost function landscape:

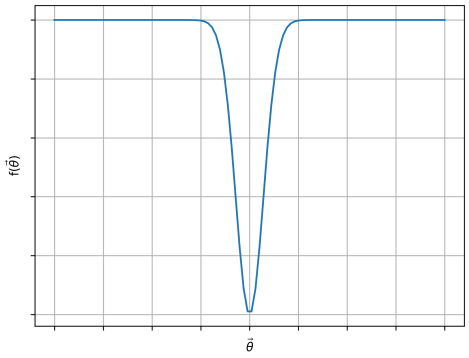


Figure 3.4.: Visualization of a barren plateaus

The plateaus left and right of the minimum are called barren plateaus [38]. These areas have to be avoided when training a quantum machine learning model. Firstly, even when having exact gradients, it is extremely slow to get out of these regions and reach the minimum. Secondly, when using SPSA, the gradients are only approximated, and it is unsure if the model even can get out of the barren plateau.
Avoiding barren plateaus is a current research question in quantum machine learning [45]. Literature explores methods for mitigating those problems. A method to do so seems to be layer-wise training with variational circuits. That means that in the course of optimization, the layers of the model increase incrementally and in each step, only a subset of parameters is optimized [58]. In general, the parameters of variational circuits are initialized randomly. Another recent advancement in quantum machine learning suggests the use of classical neural networks for the initialization of the parameters [15]. In [15] the authors propose the usage of parameters for variational circuits from classical neural networks. They show that their method mitigates barren plateaus [15].

4. A short introduction to category theory

Category theory is a modern branch of mathematics, that understands objects through their relationship with other objects [26]. These relationships are called morphisms. With consistent execution of this main principle, one can show similarities between any kind of mathematical object and show general principles across the field of mathematics [26]. In this section, we will give a short introduction to this field of mathematics and describe a few important concepts. For a more complete description of the field, we like to refer the interested reader to [32] and [36]. Category theory is extensively needed in this thesis since the DisCoCat framework in chapter 5 is categorical by design. As of right now, the DisCoCat framework is the main framework in the field of quantum natural language processing.

4.1. Categories

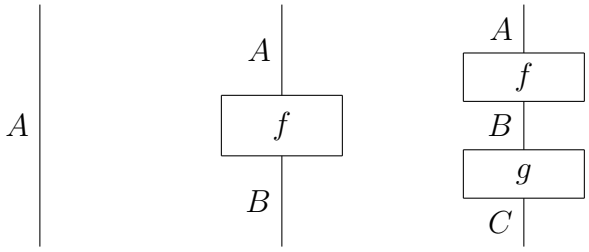
The basic elements of category theory are categories.

Definition 4.1.1. A category $\boldsymbol{C}$ is defined as [26]:

- A collection of objects A,B,C,D,... in $\boldsymbol{C}$ together with a collection of morphisms $f : A \to B$, $g : B \to C$, $h : C \to D$,... in $\boldsymbol{C}$
- Compatible morphisms can be composed. The composition will be: $g \circ f : A \to C$
- The composition of morphisms is associative: $(g \circ f) \circ h = g \circ (f \circ h)$
- For every object A, there is an identity morphism: $id_A : A \to A$, such that for every $f : A \to B$ holds: $f \circ id_A = f = id_B \circ f$

The core idea is, that elements of a category belong together because they have a similar structure. This structure needs to be maintained with morphisms [26]. Graphically, one

can draw objects as wires and morphisms as boxes. The next picture shows examples of the graphical notation. The left drawing shows the single identity morphism on object A. Also consider $f : A \to B$ and $g : B \to C$ to be morphisms. The composition of processes is then drawing the boxes in a logical sequence. The middle picture shows one process being applied, whereas the right picture shows a composition of morphisms.



For the remainder of this thesis, if objects in a category $\boldsymbol{C}$ are mentioned it is of notation A in $obj(\boldsymbol{C})$. The same goes for morphisms: f in $mor(\boldsymbol{C})$.

For later purposes, it is helpful to think of objects of a specific category as a system and morphisms as processes, which turn a system A into a system B, all while keeping the underlying structure [20]. A handy example of such a description would be in computer science with data types as systems and algorithms as morphisms, where data types get manipulated by algorithms [20].

4.2. Functors

The next crucial concept of category theory is a functor. When one wants to map a category $\boldsymbol{C}$ to a category $\boldsymbol{D}$ one intuitively needs to map objects to objects and morphisms to morphisms. Those mappings are called functors [26]. For those functors the following rules hold, let F denote the functor [26]:

- Each object X in $obj(\boldsymbol{C})$ gets mapped to an object $F(X)$ in $obj(\boldsymbol{D})$
- Each morphism $f : X \to Y$ in $mor(\boldsymbol{C})$ gets mapped to a morphism $F(f) : F(X) \to F(Y)$ in $mor(\boldsymbol{D})$ such that:
  - $F(id_X) = id_{F(X)}$ for every X in $obj(\boldsymbol{C})$; Functors preserve the identity morphism

– $F(f \circ g) = F(f) \circ F(g)$ for all $f : X \to Y$ and $g : Y \to Z$ in $mor(\boldsymbol{C})$; Functors preserve the composition of morphisms

Consider the following commutative diagram, which explains the preservation of composition [20] [5]:

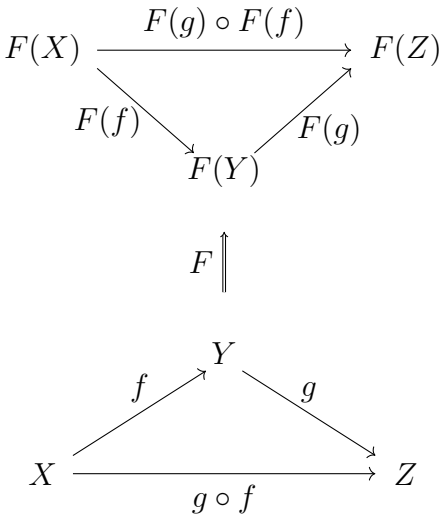


Figure 4.1.: Functor

The motivation behind a functor can be to map the structures of a complicated category to a simpler category. Another motivation can be to map structures of the category, in which no complex calculations can be performed, to another category, where there is a possibility to do so. The latter motivation will be explored in chapter 5.

4.3. Monoidal categories

The notion of monoidal categories introduces the concept of parallelization. In short, if a category is monoidal, we could let processes, i.e. morphisms, run in parallel [20]. We first introduce the category $\boldsymbol{C} \times \boldsymbol{C}$:

- The objects of category $\boldsymbol{C} \times \boldsymbol{C}$ are all possible pairs of objects (X, Y), where X and Y are objects from category $\boldsymbol{C}$.
- The morphisms of $\boldsymbol{C} \times \boldsymbol{C}$ are defined as pairs of morphisms. If there is a morphism $f : A \to B$ in $mor(\boldsymbol{C})$ and $g : C \to D$ in $mor(\boldsymbol{C})$, then the morphisms of $\boldsymbol{C} \times \boldsymbol{C}$ are of the form $(f, g) : (A, C) \to (B, D)$.

Definition 4.3.1. A monoidal category C is a category with the following additional properties [20]:

- A tensor product functor $\otimes : \boldsymbol{C} \times \boldsymbol{C} \to \boldsymbol{C}$;
  Such that all object pairs (X, Y) in $obj(\boldsymbol{C} \times \boldsymbol{C})$ get mapped to a composite $X \otimes Y$ and all morphism pairs f, g in $obj(\boldsymbol{C})$, where $f : A \to B$ and $g : C \to D$, get mapped to a parallel composition $f \otimes g : A \otimes C \to B \otimes D$. This functor is also called bifunctor [47].
- A unit object I in $obj(\boldsymbol{C})$.
- A natural isomorphism α called the associator:
  For every A, B, C in $obj(\boldsymbol{C})$ it holds $(A \otimes B) \otimes C \xrightarrow{\alpha_{A,B,C}} A \otimes (B \otimes C)$
- A natural isomorphism λ called the left unitor:
  For every A in $obj(\boldsymbol{C})$ it holds $I \otimes A \xrightarrow{\lambda_A} A$
- A natural isomorphism ρ called the right unitor:
  For every A in $obj(\boldsymbol{C})$ it holds $A \otimes I \xrightarrow{\rho_A} A$

If the associator and the two unitors are identities, the monoidal category is strict. Definition 4.3.1 has to fulfill the triangle equation [47] [20]:

$$\text{Every } A, B \text{ in } obj(\boldsymbol{C}), \qquad \rho_A \otimes id_B = (id_A \otimes \lambda_B) \circ \alpha_{A,I,B}$$

and the pentagon equation [47] [20]:

$$\text{Every } A, B, C, D \text{ in } obj(\boldsymbol{C}),$$

$$\alpha_{A,B,C\otimes D} \circ \alpha_{A\otimes B,C,D} = (id_A \otimes \alpha_{B,C,D}) \circ \alpha_{A,B\otimes C,D} \circ (\alpha_{A,B,C} \otimes id_D)$$

Due to the abstract nature of these equations, it is very beneficial to look at the diagrammatic representations of these equations:

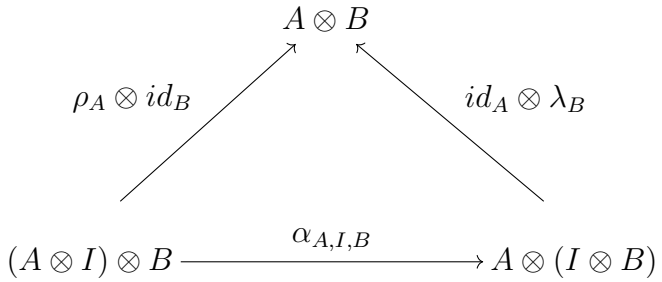


Figure 4.2.: The triangle equation

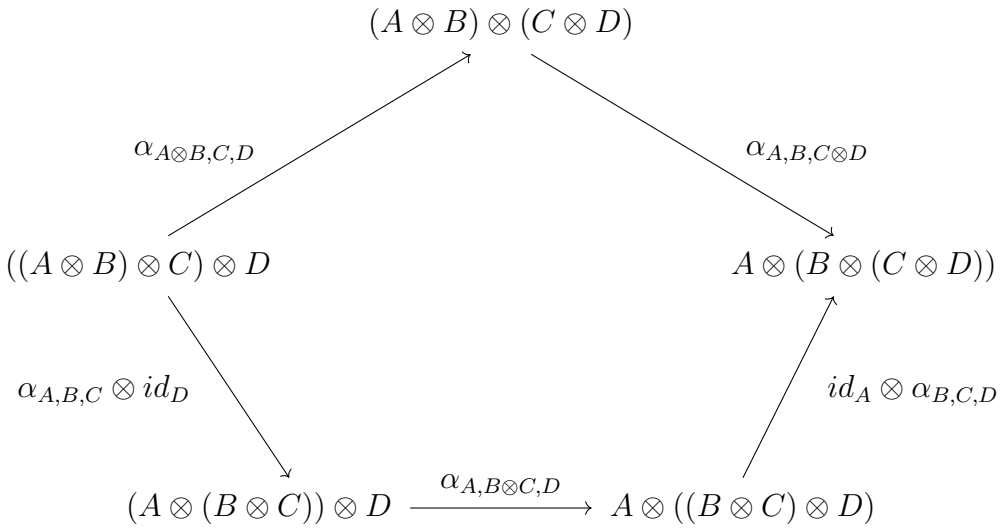


Figure 4.3.: The pentagon equation

4.3.1. Symmetry and braiding

Two important properties of monoidal categories are braiding and symmetry. In quantum computing, symmetry enables the switching of wires in quantum circuits.

Definition 4.3.2. A monoidal category is braided, if it is equipped with the natural isomorphism $\sigma_{A,B} : A \otimes B \to B \otimes A$. This morphism is called the braid, which needs to satisfy the following equations [20] [47]:

$$\alpha_{B,C,A} \circ \sigma_{A,B\otimes C} \circ \alpha_{A,B,C} = (id_B \otimes \sigma_{A,C}) \circ \alpha_{B,A,C} \circ (\sigma_{A,B} \otimes id_C)$$

$$\alpha^{-1}_{C,A,B} \circ \sigma_{A\otimes B,C} \circ \alpha^{-1}_{A,B,C} = (\sigma_{A,C} \otimes id_B) \circ \alpha^{-1}_{A,C,B} \circ (id_A \otimes \sigma_{B,C})$$

These equations are called hexagon equations. Once again, the reader is advised to look at those two equations in graphical notation:

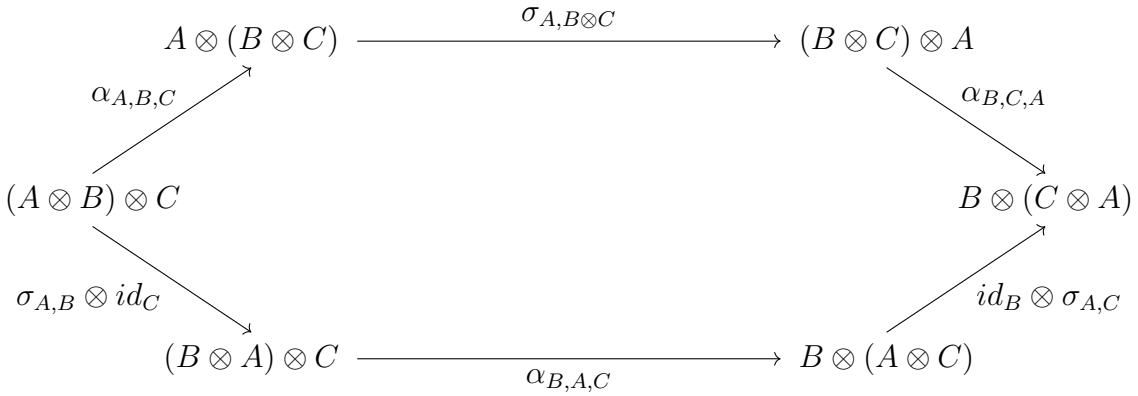


Figure 4.4.: First hexagon equation

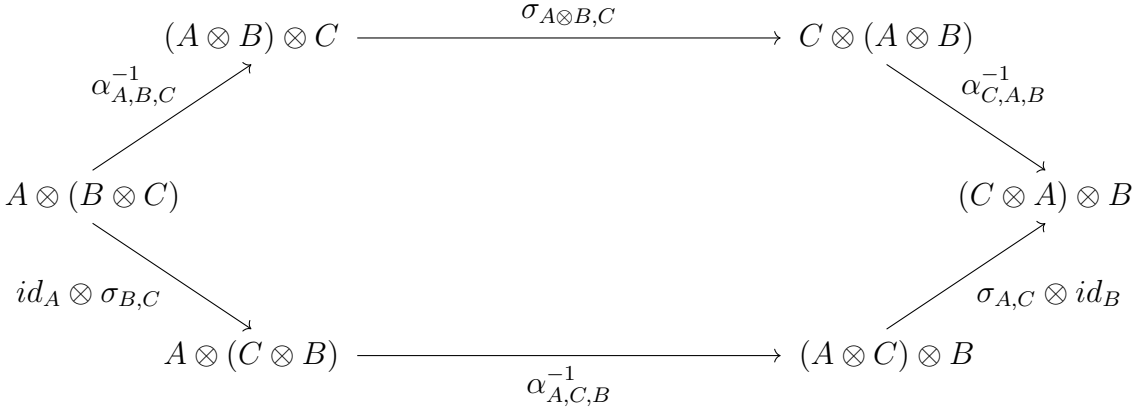
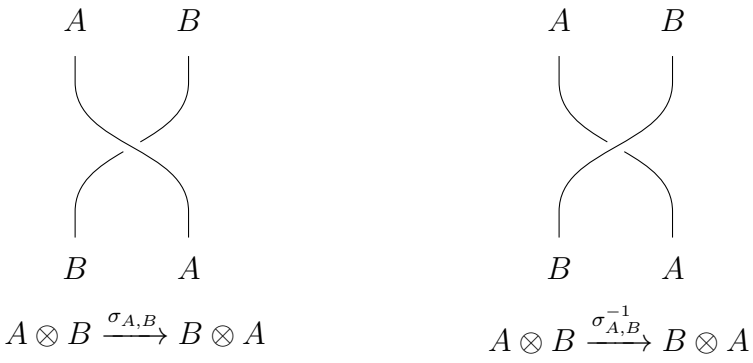


Figure 4.5.: Second hexagon equation

The braid can be represented as switching wires, depicted graphically as interrupted lines and straight lines [21]:

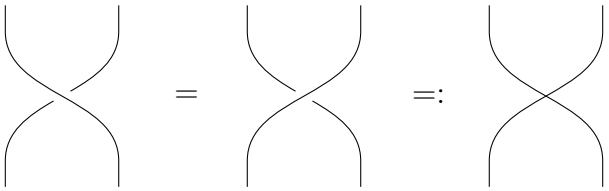


$\sigma \circ \sigma^{-}1$ and $\sigma^{-}1 \circ \sigma$ can be shown as unbraiding the wires:

In symmetric monoidal categories the braid morphism is self-inverse:

$$\sigma_{B,A} \circ \sigma_{A,B} = id_{A \otimes B}$$

Therefore, $\sigma_{A,B} = \sigma_{A,B}^{-1}$ [21]. In a symmetric monoidal category, it is therefore possible to simplify the diagrammatic representation of the braiding [21]:



4.3.2. States and effects

A way to initialize a non-trivial system is to simply bring the system into being from the trivial system I [20]. This notion gives rise to a special kind of morphism in monoidal categories called the state.

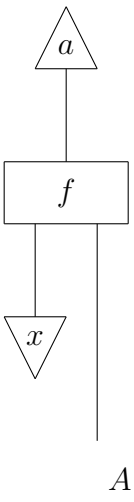
Definition 4.3.3. A state of an object A in a monoidal category is the morphism $\rho : I \to A$ [20].

Since systems can be brought into existence, the logical conclusion is that they can be also taken out of existence. These processes are called effects [20].

Definition 4.3.4. An effect of an object A in a monoidal category is the morphism $\epsilon : A \to I$ [20].

The figure below depicts a diagrammatic representation of a state on the left and its corresponding representation of an effect on the right:

The notion of an effect can be thought of as a measurement with a desired outcome. In addition, an effect can be thought of as a postselection with a desired measurement result. Consider the following diagram from [20]:



Given this diagram, a state a is first initialized. Afterwards, a process f is applied with two outcomes. The first outcome to the left is being measured with the measurement x expected. This does not mean that x is the only possible outcome to the measurement on the left, but that diagram does only take into account those instances, where x occurs. So, for example, if more processes are being applied on A afterwards, the diagram does only take into account cases, where x was measured beforehand [20].

4.4. Monoidal functors

Monoidal functors are functors between two monoidal categories that preserve monoidal structure [20]. In particular, they respect the monoidal unit and tensor product [47].

Definition 4.4.1. Let $\boldsymbol{C}$ and $\boldsymbol{C'}$ be monoidal categories. For the monoidal functor $F : \boldsymbol{C} \to \boldsymbol{C'}$ there are the following two maps [20] [27]:

$$\phi_{A,B} : F(A) \otimes F(B) \to F(A \otimes B)$$
$$\phi : I \to F(I)$$

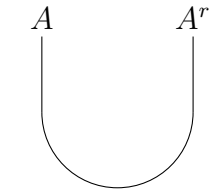
A monoidal functor is strong if $\phi_{A,B}$ and ϕ are invertible.

4.5. Compact closed categories

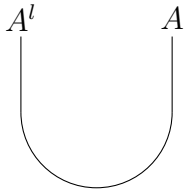
A monoidal category is compact closed if every object A in $obj(\boldsymbol{C})$ has a left adjoint A^l and a right adjoint A^r, while the following morphisms hold [27] [47]:

$$\epsilon^r_A : A \otimes A^r \to I \qquad \eta^r_A : I \to A^r \otimes A$$
$$\epsilon^l_A : A^l \otimes A \to I \qquad \eta^l_A : I \to A \otimes A^l$$

The ϵ_A morphisms can be diagrammatically represented as:

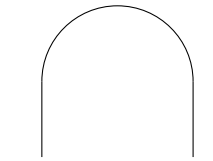


$\epsilon^r_A : A \otimes A^r \to I$

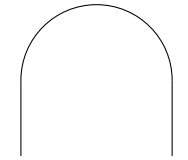


$\epsilon^l_A : A^l \otimes A \to I$

And the η_A morphisms can be diagrammatically represented as:



$\eta^r_A : I \to A^r \otimes A$



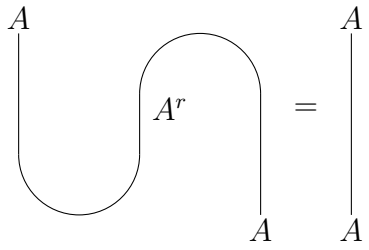
$\eta^l_A : I \to A \otimes A^l$

These structures are called cups and caps respectively [47]. Cups can be also thought of as contracting or reducing systems or part of systems. In the ϵ_A morphism case, structures contract if the right adjoint is matched.

In addition, the η_A and ϵ_A morphisms are needed in compact closed categories so that the following equations hold [20]:

$$(\epsilon^r_A \otimes id_A) \circ (id_A \otimes \eta^r_A) = id_A \qquad (id_{A^r} \otimes \epsilon^r_A) \circ (\eta^r_A \otimes id_{A^r}) = id_{A^r}$$
$$(id_A \otimes \epsilon^l_A) \circ (\eta^l_A \otimes id_A) = id_A \qquad (\epsilon^l_A \otimes id_{A^l}) \circ (id_{A^l} \otimes \eta^l_A) = id_{A^l}$$

These equations are called yanking equations or snake equations, due to their form. To illustrate a yanking equation, consider the following diagram, which corresponds to the top-left yanking equation [47]:



Visually, the diagram on the left gets stretched or yanked straight, such that the id_A morphism gets produced, hence the name.

4.6. FHilb category

The finite-dimensional Hilbert Spaces, which were discussed briefly in section 3.1, form a compact closed category [12]. For complete mathematical proof, the reader is referred to [11]. This compact closed category will be further referenced as FHilb. In FHilb objects are Hilbert spaces, morphisms are linear maps, composition is the composition of linear maps, and the identity morphism is the identity linear map [20]. The monoidal product is the tensor product [27]. As this compact closed category is symmetric, we observe the following property. Remember V^l and V^r are the left and right adjoint respectively, for any object V in $obj(FHilb)$:

$$V^l = V^r = V^*$$

V^* is called the dual. By fixing a basis $\{\vec{r}_i\}_i$, the dual becomes isomorphic to the vector space itself, thus $V^* \cong V$ [27]. The ϵ and η look the following way [27] [12]:

$$\epsilon_V^r = \epsilon_V^l : V \otimes V \to I \quad \text{given by} \quad \sum_{i,j} c_{i,j} \vec{v}_i \otimes \vec{w}_j \mapsto \sum_{i,j} c_{i,j} \langle \vec{v}_i | \vec{w}_j \rangle$$
$$\eta_V^r = \eta_V^l : I \to V \otimes V \quad \text{given by} \quad 1 \mapsto \sum_i \vec{r}_i \otimes \vec{r}_i$$

With this information, the yanking equations can be solved. Note that due to the property of this category the four yanking equations collapse into the following two:

$$(\epsilon_A \otimes id_A) \circ (id_A \otimes \eta_A) = id_A$$
$$(id_A \otimes \epsilon_A) \circ (\eta_A \otimes id_A) = id_A$$

For an example proof of a yanking equation, the reader is referred to section 3.1.2 in [47].

5. Categorical compositional distributional framework

The categorical compositional distributional framework, also known as DisCoCat, is a mathematical approach to natural language processing incorporating aspects of category theory. It was first proposed by Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark in 2010 [12]. In this section, we will comprehensively describe this idea and bring it into the context of quantum computing. This framework is of big importance in this thesis since it is the foundation of current quantum natural language processing.

5.1. Pregroup grammar

An important prerequisite to DisCoCat is the pregroup grammar introduced by Joachim Lambek [31]. A Lambek pregroup is denoted by a six tuple $(P, 1, \cdot, \leq, -^r, -^l)$, which is a partially ordered algebra [27]. Each element of this partial order has a right and left adjoint. This algebra forms a compact closed category, such that the yanking equations for a partial order hold [55]:

$$p \cdot p^r \leq 1 \leq p^r \cdot p \qquad\qquad p^l \cdot p \leq 1 \leq p \cdot p^l$$

The morphisms defining compact closed categories also hold:

$$\begin{aligned} \epsilon^r &: p \cdot p^r \leq 1 & \qquad \eta^r &: 1 \leq p^r \cdot p \\ \epsilon^l &: p^l \cdot p \leq 1 & \qquad \eta^l &: 1 \leq p \cdot p^l \end{aligned}$$

Whereas ϵ morphisms are sometimes called contractions and η morphisms extensions. Note that $\cdot$ and $\leq$ can be written as $\otimes$ and $\rightarrow$. For the application of the pregroup grammar on natural language, it is necessary to define the elements for the pregroup grammar, i.e. the elements in P. Such elements are often called types or atomic types [28]. These types can be thought of as atomic linguistic structures that can occur in

a sentence. For example, simple types are the noun type n, the sentence type s, and the prepositional phrase type p. After defining the types, a lexicon of words L and a function f that maps each word to a type or combination of types are needed. Let the following simple example illustrate this idea:

$$
\begin{aligned}
P &= [n, s] \\
L &= [Alice, cooks, eats, schnitzel, at, home, Bob, loves] \\
f &= \{Alice : n, cooks : n^r \cdot s \cdot n^l, eats : n^r \cdot s \cdot n^l, schnitzel : n, at : s^r \cdot n^{r^r} \cdot n^r \cdot s \cdot n^l, \\
&\qquad home : n, Bob : n, loves : n^r \cdot s \cdot n^l\}
\end{aligned}
$$

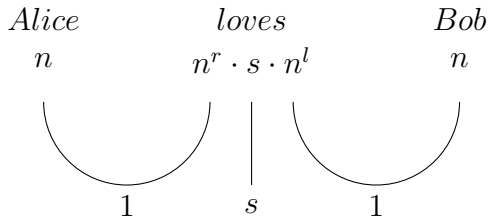
We now look at the sentence *Alice cooks schnitzel at home.*:

$$
\begin{aligned}
n \cdot n^r \cdot s \cdot n^l \cdot n \cdot s^r \cdot n^{r^r} \cdot n^r \cdot s \cdot (n^l \cdot n) &\leq \\
(n \cdot n^r) \cdot s \cdot n^l \cdot n \cdot s^r \cdot n^{r^r} \cdot n^r \cdot s \cdot 1 &\leq \\
1 \cdot s \cdot (n^l \cdot n) \cdot s^r \cdot n^{r^r} \cdot n^r \cdot s \cdot 1 &\leq \\
1 \cdot (s \cdot 1 \cdot s^r) \cdot n^{r^r} \cdot n^r \cdot s \cdot 1 &\leq \\
1 \cdot 1 \cdot 1 \cdot (n^{r^r} \cdot n^r) \cdot s \cdot 1 &\leq \\
1 \cdot 1 \cdot 1 \cdot 1 \cdot s \cdot 1 &= s
\end{aligned}
$$

It is shown that with this grammar the sentence contracts to the atomic type s. When sentences reduce to the atomic type s the sentences are well typed [12]. Note that there are different ways of contracting the types. In this example, a simple approach is used for illustration purposes, but an automatic parser might choose a different way. The next example with *Alice cooks eats.* shows a sentence that is not well-typed and therefore does not reduce to the s type:

$$
\begin{aligned}
(n \cdot n^r) \cdot s \cdot n^l \cdot n^r \cdot s \cdot n^l &\leq \\
1 \cdot s \cdot n^l \cdot n^r \cdot s \cdot n^l &
\end{aligned}
$$

No more reduction can be performed after this first step and the sentence does not reduce to type s. As mentioned beforehand, diagrammatically contraction can be depicted as cups. Consider the sentence *Alice loves Bob*:



It is also diagrammatically shown, that the sentence *Alice loves Bob* is well typed.
Since the pregroup grammar needs a lexicon and an according mapping with all lexicon entries to some types, it is a challenge to implement such approaches to real-life examples. First, out-of-vocabulary words pose a problem. Secondly, the biggest challenge is to find such needed mapping. In the Lambeq package [28] these challenges are addressed by using a large language model as a parser. The Lambeq package includes the BobCat parser, which has a Transformer-based architecture. It syntactically parses every sentence, such that it can be represented in a pregroup grammar. Note that because of the neural network based architecture, every process inside this parser is of statistical nature. For a more detailed description of how this parser operates, the reader is referred to the original paper [9].

5.2. The meaning of words as vector space model

The second important prerequisite to the DisCoCat model is how the meaning of words is modeled in computational linguistics. The core idea behind a vector space model for meaning can be explained by the well-known quote by John R. Firth "you shall know a word by the company it keeps" [12]. That means the meaning of a word is represented by other words inside its context. The context can be a window of any size. That means that for example the words *tree* and *bush* have similar meanings because both appear in similar contexts. Neighboring words of both examples might be *green*, *ground*, *bloom*, and many more. Thus, a big part of the contexts of *tree* and *bush* overlap and give those two similar meanings. It is important to say that not everything in the two contexts overlap since they are both still two separate entities. *tree* might be mentioned alongside *tall*, whereas *bush* might be alongside *tiny*. The same ideas go for words that are completely different from each other. *car* and *human* will have only a small amount of context words that overlap. All of these intuitions are reflected in natural language texts, which humans produce every day [12].
The meaning of words is then represented as a high-dimensional vector $\vec{v} \in \mathbb{R}^n$ in a

meaning-space. The orthogonal basis vectors are then the possible words that can occur in context [12]. To explain this concept further consider this simple example. The following context words define the basis: *green*, *tall*, and *machine*. Imagine we want to represent the words *tree*, *bush*, and *car*. In a given corpus suppose the following co-occurrence count:

	context word *green*	context word *tall*	context word *machine*
tree	5	7	0
bush	5	4	1
car	2	1	7

Table 5.1.: Co-occurrence count

This matrix presents the following vectors: $\vec{v}_{tree} = (5, 7, 0)$, $\vec{v}_{bush} = (5, 4, 1)$ and $\vec{v}_{car} = (2, 1, 7)$. Note that in more serious applications the co-occurrence number is usually normalized through tf-idf (term frequency inverse document frequency) or other similar measures [27]. By defining a vector space there is the opportunity to calculate the distance of vectors. This can be done through various metrics such as the inner product or the cosine similarity. The cosine similarity of two n-dimensional vectors $\vec{a}$ and $\vec{b}$ is of the following form:

$$CS(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{||\vec{a}|| \cdot ||\vec{b}||}$$

The distance of the word vector enables the notion of similarity between words. The closer the vectors the more similar the words. In terms of cosine similarity that means the closer the value to 1 the closer the vectors and therefore the more similar the words. We can experience this phenomenon in the example above. $CS(\vec{v}_{tree}, \vec{v}_{bush}) \approx 0.95$ and $CS(\vec{v}_{tree}, \vec{v}_{car}) \approx 0.27$. *tree* is more similar to *bush* than to *car*. Vector spaces can therefore carry the meaning of words.
Vector-based approaches have been successfully applied to many aspects of natural language processing [12] such as text segmentation [8], information retrieval [46] [42], automated word sense discrimination [54], and prominently, language modeling [4].

5.3. The meaning of a sentence

We have seen so far that the pregroup grammar, i.e. the category of grammar, and FHilb, i.e. the category of meaning, both form a compact closed category. We also

saw that conveniently every quantum state in quantum computing lies inside a finite dimensional Hilbert space.

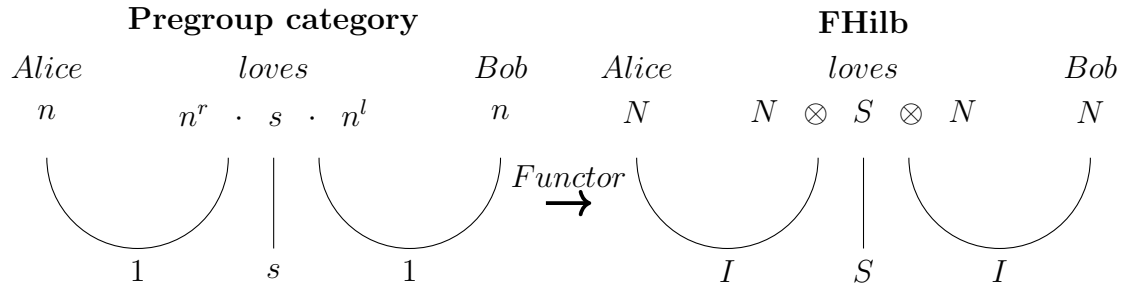
Bob Coecke et al. [12] propose a strong monoidal functor that maps the grammar category to the meaning category. By mapping the grammar of a sentence to the distributional meaning of words we get the meaning of a sentence composed by the meaning of words combined with the grammatical structure [12]. This idea is made possible by category theory.

Recall that a functor maps objects to objects and morphisms to morphisms while keeping the categorical structure. We start with the objects. As presented earlier in section 5.1 the grammar category holds several atomic linguistic types (such as n_{noun}, $s_{sentence}$, $p_{prepositionalphrase}$ etc.). The FHilb category holds as objects Hilbert spaces. We map the atomic linguistic types to the finite-dimensional Hilbert spaces. We have the freedom to decide with how many qubits the Hilbert space is represented with, that the atomic linguistic type gets mapped to. This will later be mentioned as a hyperparameter in the quantum machine learning model. Note that while mapping $s_{sentence}$ to S, as S corresponds to a d-dimensional Hilbert space, it is also necessary to map right and left adjoints to the Hilbert space. As FHilb is symmetric and therefore has a dual object, which is isomorphic to itself, it holds that when mapping $s^r_{sentence}$ to S^r: $S^r = S^* = S$. Thus, $s^r_{sentence}$ gets mapped to S. Every adjoint of a type in the grammar category gets mapped to one self-isomorphic object in the FHilb category.

The morphisms in the pregroup category, i.e. the contractions ϵ^r and ϵ^l and extensions η^r and η^l, are then mapped to contraction and extension in FHilb. Note again that since finite vector spaces are self-dual there is only one contraction morphism and one extension morphism. It looks the following way, whereas $\mapsto$ depicts the mapping and the subscript depicts the according category:

$$\epsilon^r_P \mapsto \epsilon_{FHilb} \qquad \epsilon^l_P \mapsto \epsilon_{FHilb}$$
$$\eta^r_P \mapsto \eta_{FHilb} \qquad \eta^l_P \mapsto \eta_{FHilb}$$

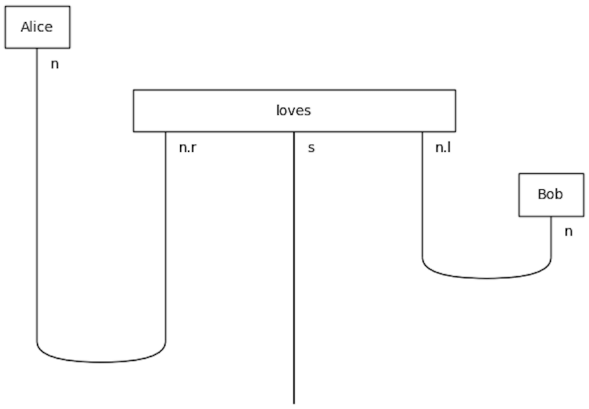
Diagrammatically the process of mapping one category to the other is depicted as:



It is important to note that not just the pregroup grammar introduced by Joachim Lambek [31] works but any grammar, which can be represented as a compact closed category, works[47].

Note that in the FHilb category, which is informed by the grammar category, every group of objects carries the meaning of a word. That means the first group N in the example above carries the meaning of the word *Alice*, $N \otimes S \otimes N$ carries the meaning of *loves*, and the last group N carries the meaning of *Bob*. These groups of objects are later represented as boxes.

The structure in FHilb is determined by the grammar category, while the meaning itself gets left alone. If we then perform the reduction morphisms of this compact closed category, we get left with the S wire. This wire carries the meaning of the sentence, where semantics and syntax are combined [12]! We can think of this whole procedure as one state $Alice\ loves\ Bob : I \otimes I \otimes I \to S$, where S carries the representation of the entire diagram [47]. In the Lambeq package [28] the pregroup diagrams are often combined with the aforementioned word boxes, where each word box will be mapped to its corresponding meaning in the FHilb category. This diagram style is helpful for illustration purposes:



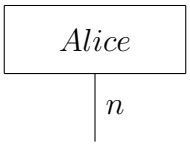
Note that in those sorts of diagrams, the time evolution is depicted from top to bottom.

5.4. From FHilb to quantum circuits

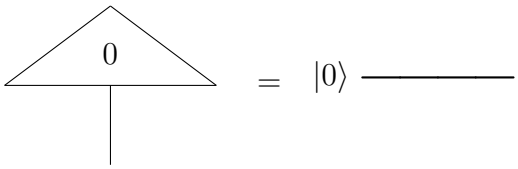
Since FHilb can be represented in terms of quantum theory, the grammar-enriched FHilb category can be transformed into a quantum circuit. This transformation is one of the most challenging concepts in this framework. This transformation will be explained step by step, starting with the simpler systems.

5.4.1. Initialization

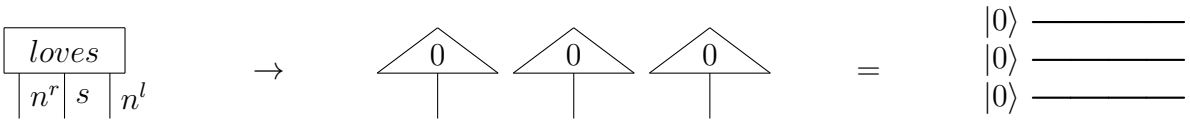
As explained in section 4.3.2, we first need to bring a state or system into existence. This is achieved by the morphism $\rho : I \to A$. Consider the isolated system for *Alice*:



According to section 4.3.2, this will be transferred to:



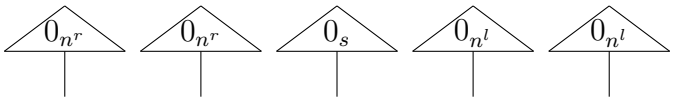
This pipeline also holds for boxes with a higher codomain:



With this representation, there exists a degree of freedom in the choice of the number of qubits for each object. Remember, since FHilb is symmetric and by fixing a basis, the dual becomes isomorphic to the vector space itself every object plus its adjoints gets mapped to one object in FHilb.

$$n \mapsto N; \;\; n^r \mapsto N; \;\; n^l \mapsto N$$

We are free to choose the number of qubits for N. This choice will be later handled as a hyperparameter. If we consider the *loves* box from above, where we map to a Hilbert space, where S is represented by one qubit and N is represented by two qubits, the initialized state for *loves* looks like:



5.4.2. Addition of meaning

As discussed in section 5.2, meaning can be represented in terms of vectors. The natural question that arises is, how can one implement those meaning-vectors inside a quantum circuit? In section 3.5.2 circuits with parameters were explained. The parameters of the rotation gates in the parameterized circuits represent the meaning of the word. In source [40] it is described as using the quantum parameters as word embeddings. The Hilbert space is used as feature space [40].

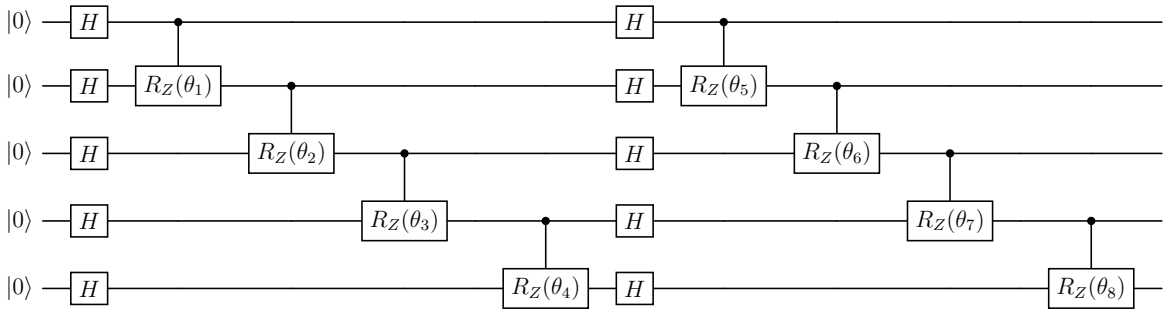
Single qubit cases

For representing the meaning of a box that is mapped to one qubit it is commonly practiced adding rotation gates in the form of $R_z(\theta)R_x(\theta)R_z(\theta)$ [40]. This sequence of rotation gates is important since it allows us to represent any state on the Bloch sphere. As explained in section 3.3.1 it is possible to decompose any unitary operation on one qubit into MNM rotation, where M and N are two non-parallel rotation axes. We strongly want to cover the entire Bloch sphere since expressibility of quantum circuits increases learning capability, as explained in section 3.5.2.
To summarize, the meaning-vector of a single qubit box is represented as $R_z(\theta)R_x(\theta)R_z(\theta)$ rotation procedures. If we want to have a meaning-vector of size nine, we have three blocks of those rotation procedures. A word vector with fewer than three gates is not optimal.

Multi qubit cases

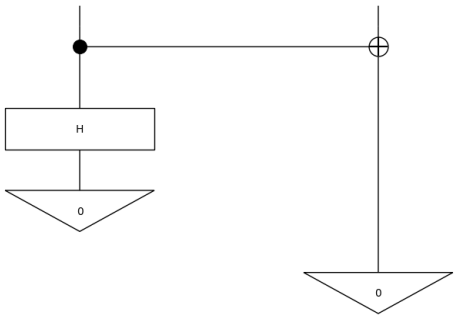
In the majority of times, boxes have more than one output wire. In section 3.5.2 various subroutines of choosing the gates for variational circuits are explained. These subroutines, or ansätze, are used to represent multi qubit boxes. In section 3.5.2 one layer

of such ansätze is described, but in a machine learning context, one can choose how many layers of an ansatz are being used. This choice will also be later introduced as a hyperparameter.
A simple example for the *loves* from above: We map N to 2 qubits, S to 1 qubit, and have two layers of the IQP ansatz. The quantum circuit looks like this:



5.4.3. Cups and caps in quantum theory

The most difficult concept of this mapping is the representation of the caps and cups in quantum computing. Recall that in FHilb caps and especially cups are of utmost importance.
In [40] the authors propose to map cups to:



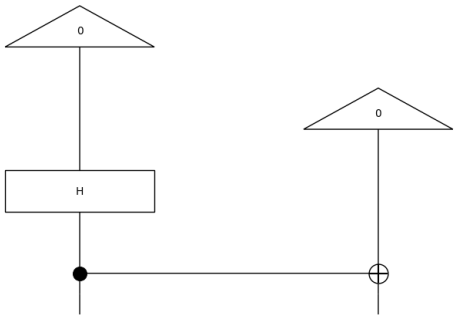
The output state of the cup follows as:

$$\begin{aligned}(\langle 0| \otimes \langle 0|)(H \otimes I)(|0\rangle \langle 0| I \otimes |1\rangle \langle 1| X) = \\ \frac{1}{\sqrt{2}}((\langle 0| + \langle 1|) \otimes \langle 0|)(|0\rangle \langle 0| I \otimes |1\rangle \langle 1| X) = \\ \frac{1}{\sqrt{2}}(\langle 00| + \langle 10|)(|0\rangle \langle 0| I \otimes |1\rangle \langle 1| X) = \\ \frac{1}{\sqrt{2}}(\langle 00| + \langle 11|)\end{aligned}$$

Hence:

$$\cup = \frac{1}{\sqrt{2}}(\langle 00| + \langle 11|)$$

Caps are proposed to be:



The output state of the cup follows as:

$$\begin{aligned}(|0\rangle \langle 0| I \otimes |1\rangle \langle 1| X)(H \otimes I)(|0\rangle \otimes |0\rangle) = \\ (|0\rangle \langle 0| I \otimes |1\rangle \langle 1| X)(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle) = \\ (|0\rangle \langle 0| I \otimes |1\rangle \langle 1| X)(\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)) = \\ \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)\end{aligned}$$

Hence:

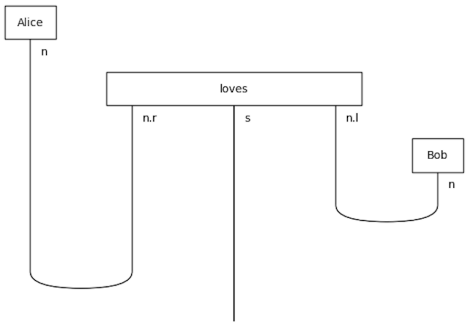
$$\cap = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Recall that in FHilb the four yanking equation collapse into just two. With the aforementioned cap and cup representation we can verify the yanking equations. For the full

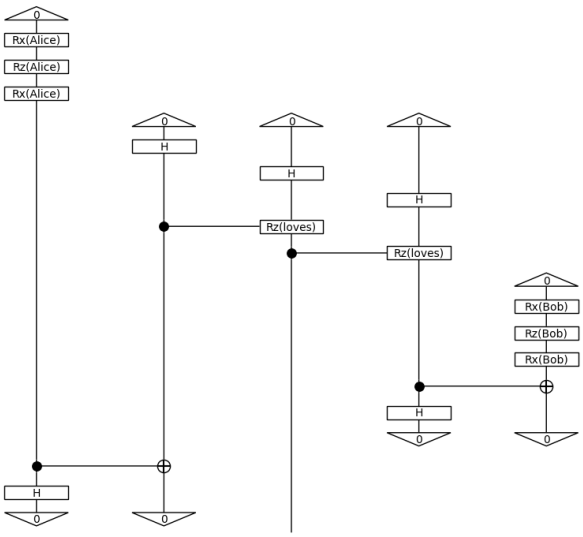
verification we refer the reader to appendix G.

5.4.4. The full picture

Having described all the necessary representation, it is now possible to show the entire pipeline from a sentence to a trainable quantum circuit. We do this by mapping every basic type in the pregroup grammar to one qubit. We are also using three single qubit rotation gates and one layer of the IQP ansatz. The example sentence will be *Alice loves Bob*.
Starting with the grammatical representation:



Then continue with the according circuit. The non-classical representation of quantum circuits from [28] will be used since they are more complete in terms of postselection:



In those circuit representations, the information flows from top to bottom. The first qubit is on the left.
At last, the entire DisCoCat framework is comprehensively mathematically described by Sadrzadeh et al. in [51]:

Definition 5.4.1. The DisCoCat framework can be depicted by a 4-tuple:

$$(\boldsymbol{C}_{Syn}, \boldsymbol{C}_{Sem}, F, [\![\,]\!])$$

This 4-tuple consists of a compact closed category $\boldsymbol{C}_{Syn}$ and $\boldsymbol{C}_{Sem}$. It contains a strongly monoidal functor $F : \boldsymbol{C}_{Syn} \to \boldsymbol{C}_{Sem}$ and a semantic map $[\![\,]\!] : \Sigma^* \to \boldsymbol{C}_{Sem}$, that maps every set of strings from a language Σ^* to the category of semantics.
The meaning of phrases and sentences is therefore represented by:

$$[\![w_1 w_2 ... w_n]\!] := F(\alpha)([\![w_1]\!] \otimes [\![w_2]\!] \otimes ... \otimes [\![w_n]\!])$$

$w_1 w_2 ... w_n$ denotes the phrase and α denotes the grammatical structure of that phrase, such that α in $mor(\boldsymbol{C}_{Syn})$. On the left side of the above equation, the semantics of the sequence of strings is depicted, such that it is calculated by the single meanings of each word w_i with the help of α, F and $\otimes$.

The principle of lexical substitution is implemented in this framework. It dictates how the meaning of the words composes the meaning of the sentence. For a more detailed explanation of lexical substitution, the reader is referred to [57].

5.5. Combination with quantum machine learning

As already suggested this framework will be combined with the means of machine learning. Generally, there is no direct prescription with which machine learning methods the DisCoCat framework should be combined with. As already indicated, we will focus on quantum machine learning. Later there will be a short explanation of why quantum is a good decision for the DisCoCat framework.
As described earlier, the meaning of a sentence is represented in the output wire, which does not have a 0-effect attached to it. In a quantum theory context, this wire corresponds to a qubit that is being measured. As described in section 3, a qubit is mathematically mapped to $\mathbb{C}^2$. The output vector of a sentence in the form of $[a_1 + b_1 i; a_2 + b_2 i]$ gets transformed to $[||a_1 + b_1 i||; ||a_2 + b_2 i||] = [r_1; r_2]$ whereas $r_1, r_2 \in \mathbb{R}$ and $r_1 + r_2 = 1$,

compare section 3.2. If the DisCoCat framework is combined with a classification task for a sentence, the values r_1 and r_2 respond to our predicted labels. To retrieve predicted labels, it is commonly practiced to round the predictions to get discrete values [40]. An according loss function needs to be chosen as well as an optimizer. As explained in section 3.5.3, the optimizer poses a problem in quantum machine learning. Therefore, mostly the SPSA algorithm is being used.

5.6. Suitability for quantum computing

In the first description of the DisCoCat framework, there is no direct mention of quantum information. This fact should somewhat foreshadow the actual development of DisCoCat. The motivation behind the development of this framework was more of a practical concern in NLP: If we know the meaning of words, how can we compose the meaning of a sentence [13]? It happened to be that category theory granted the necessary ingenuity to make a framework that provided a blend between meaning and grammar [13]. Of course, there is no denying the fact that categorical quantum mechanics and DisCoCat are inseparable and that quantum computing lives inside categorical quantum mechanics. So the term, the authors prefer the most for describing the relationship between quantum computing and DisCoCat, is quantum-inspired [13]. It is also argued since DisCoCat framework is conceptualized inside a Hilbert space, albeit it is informed by the pregroup grammar, the whole framework is quantum-native, since Hilbert spaces lay the foundation in quantum computing.
The next question to ask is whether quantum natural language processing will also benefit from the quantum speedup we are seeing in quantum algorithms, for example in Grover search [18]. It is obvious, that quantum computers do not suffer from the exponential space blowup in tensor products [13]. But can it bring benefit to NLP problems? In [13] it is argued that in several standard NLP task, such as latent semantic analysis or question answering, we can achieve quantum advantage. A more immediate benefit is for example the storage of meaning-vectors by quantum computing. Each n-qubit system has 2^n degrees of freedom, which means it can store N-dimensional classical vectors using just $log_2 N$ qubits [1]. However, it is essential to approach all these theoretical quantum advantages with caution because, as of now, the scalability and noise levels in current quantum hardware pose significant challenges.

6. Experiment

As seen in chapter 2, word prediction is a commonly occurring task in pretraining language models. This forms the motivation for exploring word prediction in the DisCoCat framework. In particular, the core of the experimental part in this thesis deals with the implementation of word prediction as found in masked language modeling in a quantum machine learning setup using the DisCoCat framework. The experiment section will first deal with the related work regarding this area. Afterwards, the section will be split into two parts, one fraction will deal with an approach to masked language modeling in a binary classification setup and the other fraction will deal with a multiclass classification realization. For each implementation, the experiment setup will be thoroughly explained and evaluation will be done on unseen data. The dataset from [59] will be used. For the second multiclass classification part, theoretical approaches to a masking strategy in a quantum machine learning setup are comprehensively presented, a strategy is described how to reduce task complexity and qubit amount, and lastly, it is thoroughly explained how to do a multiclass classification in the DisCoCat framework.

6.1. Related work

In this section, some key experiments in the field of quantum natural language processing will be reviewed, focusing on previous work that has explored the use of quantum natural language processing for tasks such as density matrices for word embedding and text classification. All the following work uses the DisCoCat framework [12].

6.1.1. First QNLP experiment on quantum hardware

In [40] the authors train a quantum machine learning model on a sentence classification task. This work is the first-ever QNLP experiment on a noisy intermediate-scale quantum processor (NISQ), as discussed earlier in this thesis. The training dataset includes 30 sentences with a vocabulary size of 7 words. Each sentence was annotated with a

binary label $l_\sigma \in \{0, 1\}$ beforehand, where 0 correspond to the value False and 1 corresponds to the value True. The authors split the learning corpus into a training and a test set. The task is described by the authors as a binary classification of sentences in a supervised learning setup.
To generate a label for a sentence, they map the sentence diagrams, generated by the DisCoCat framework to pure quantum circuits. The evaluation of a circuit is a scalar $p \in [0, 1]$. By rounding to the nearest integer, the authors get the binary label. The circuits used in the experiment have variational gates which can be trained. The parameters are trained such that the predicted labels match the labels in the training set. The cost function for this task was defined as:

$$L(\theta) = \sum_{\sigma \in \Delta} (l_\sigma^{pr}(\theta_\sigma) - l_\sigma)^2$$

$l_\sigma^{pr}(\theta_\sigma)$ represents the predicted label from the model with parameters θ_σ. l_σ refers to the target label. Δ denotes the dataset and subscript σ denotes the index variable for a training sentence in the dataset.
Minimizing this cost function yields the optimal parameters for this task based on the training set. For optimization, the authors use the gradient free optimization method SPSA [50], which is described in section 3.5. The authors evaluate the performance of the model with training and test errors. Train and test errors are the number of false predictions inside the according dataset.
Firstly, they simulated their quantum circuits and received promising performance. They showed smooth convergence and simultaneously showed, that with increasing circuit depth and parameters, the training and test errors shrink. They could reduce the test error rate (percentage of wrong predicted labels on the test set) to around 0.4. For their quantum experiment on NISQ processors, they reduced their dataset to 16 sentences with a vocabulary of 6 words. They let their model run on quantum hardware from IBM (in particular ibmq montreal and ibmq toronto). They could also report smooth convergence for their quantum run and reported a test error rate of 0.375 [40]. One has to keep in mind that the corpus was significantly reduced for the quantum run. The authors stated that running QNLP models on quantum hardware in the NISQ era has concrete promise for scalability as the quality of the quantum hardware improves in the near future.
A selection of further topics given in their work is working with larger datasets and targeting more complex NLP tasks. Secondly, they suggest trying out hybrid models, where classical neural networks and pure quantum circuits are being used. At last, they pro-

pose adopting the recently introduced DisCoCirc framework [10] for QNLP experiments.

6.1.2. Classification based on meaning and grammar

In [34] the authors present two QNLP experiments, which are evaluated on quantum hardware. Like [40], both experiments are binary classification tasks in a supervised learning environment with a quantum machine learning setup.
The first proposed experiment is a meaning classification task, referenced further as MC. For a given sentence the model has to determine between two possible meaning categories, cooking or IT. The corpus for this classification has 130 sentences and each meaning category is equally represented with 65 sentences each.
The second experiment is a binary classification task, where the model has to predict whether a phrase contains a subject relative clause i.e. ("device that detects planets") or contains an object relative clause i.e. ("device that observatory has"). This task is further referenced as RP. The dataset for this experiment contains in total 105 clauses with a total of 115 words.
For later evaluation, the authors split both corpora into training, development, and test sets.
The authors employ the binary cross-entropy loss function [22] and optimize their approach using the SPSA algorithm [50]. For a description of this loss function, the reader is referred to appendix F.
As described in the experiment in [40], labels are being produced by rounding the evaluations of the circuits. The model is evaluated by calculating training and test error, like in [40]. Interestingly, the authors provide a variety of ansatz modifications they used for those two experiments. The figure 6.1 given in [34] provides a good overview of the studied hyperparameters:

MC		*RP*	
(q_s, p_n, d)	k	(q_s, p_n, d)	k
$(1, 1, 1)$	22	$(0, 1, 1)$	114
$(1, 1, 2)$	35	$(0, 1, 2)$	168
$(1, 3, 1)$	40	$(0, 3, 1)$	234
$(1, 3, 2)$	53	$(0, 3, 2)$	288

Figure 6.1.: Studied hyperparameters in [34]

q_s describes the number of qubits used for the sentence type. This parameter corresponds to circuit width. Note that for sentence evaluation the number of sentence qubits directly corresponds to the number of classes we have for the classification task with 2^{q_s}. In the MC task, there are 2 classes, hence $q_s = 1$. In the RP task there are no sentences, only clauses. Clauses reduce to the atomic type n (i.e. the meaning of the clause is represented in one leftover wire with the atomic type n), hence $q_s = 0$ and $q_n = 1$.
p_n describes how many rotation gates are used per noun. Sometimes this parameter is called "number of single qubit gates". This hyperparameter corresponds to circuit depth.
d describes the number of layers used for multi qubit gates. One layer in this experiment consists of a row of Hadamard gates and a row of controlled variational gates, that connect all qubits with each other. For more information compare section 3.5. d also corresponds to circuit depth.
At last, k describes how many parameters each ansatz with the corresponding hyperparameters has.

Firstly, the authors simulate their quantum model. With each of the proposed ansätze, they achieve a smoothly converging model. For the MC task, the model converged most optimally with hyperparameter triplet (1,3,1) and (1,3,2). For (1,3,1) the train and test error rate is at 16.9% and 20.2%, respectively. In the RP case, the model most optimally converged for the triplets (0,1,2) and (0,3,2). The train and test errors for the triplet (0,1,2) are 9.4% and 27.7%, respectively.
At last, the experiment is run on quantum hardware. The MC task is tested with the hyperparameter choice (1,3,1) and had a test error rate of 16.7%. The MC task is tested with the hyperparameter choice (0,1,2) and had a test error rate of 32.7%. As the numbers show, the quantum run yields worse results, due to noisy quantum hardware at the time.
For further work, the authors propose again larger-scale experiments, with better quantum hardware. The whole scalability question is an important topic, as quantum hardware steadily gets improved. They also outline the search for specific ansatz choices in a task-specific way. A further line of research is also investigating the introduction of regularization methods in quantum machine learning and the use of other optimization methods.

6.1.3. Density matrices as word embeddings

In [6] the author proposes a quantum machine learning model which learns density matrices as word embeddings by using Word2DM. Word2DM is a quantum approach for learning word embeddings inspired by the classical Word2Vec Continuous Skip-gram algorithm, hence the name. The Word2Vec Continuous Skip-gram algorithm dictates the model to predict the neighboring words for a selected word inside a sentence. The model does not need to predict every word in the sentence, only words given inside a certain window size. The choice of the window size is a hyperparameter. Figure 6.2 taken from [6] describes the Skip-gram objective well:

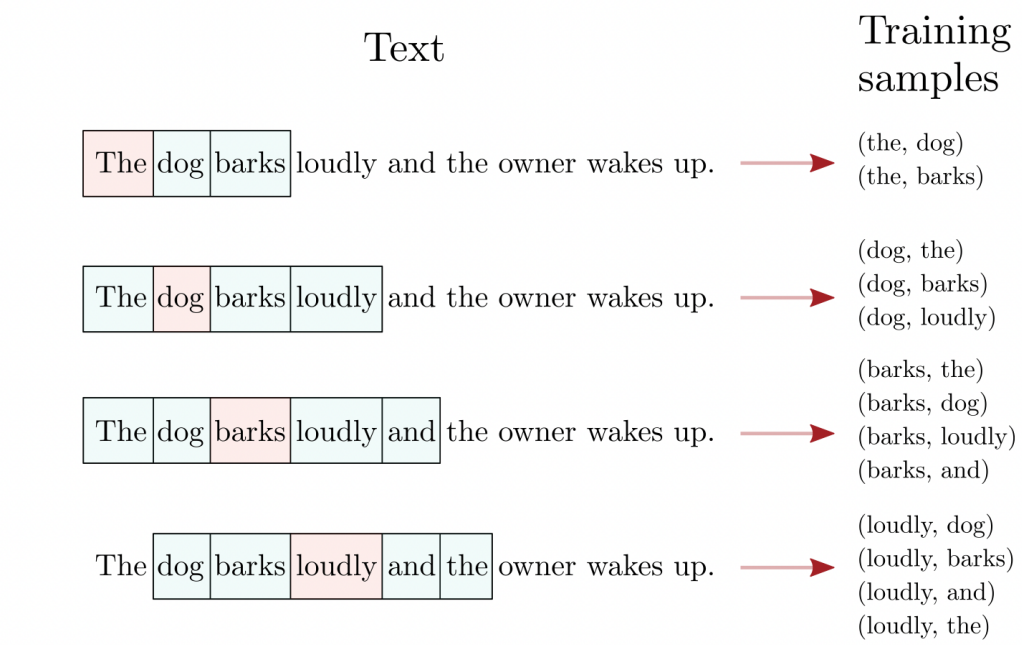


Figure 6.2.: Word2Vec Continuous Skip-gram objective with a window size of. Green boxes indicate the context words and red boxes indicate the selected word [6]

.

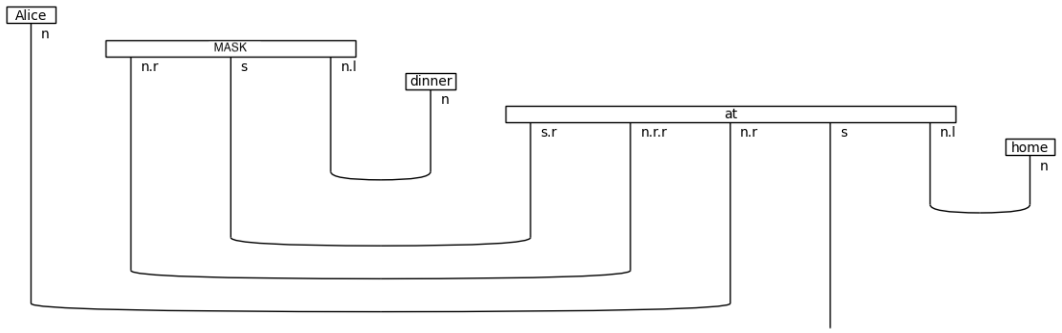
Note that in the training sample tuples depicted by the illustration, the first string represents the selected word (red boxes) and the second string indicates the word that should be predicted given the selected word (green boxes). For a more detailed description, the reader is referred to [41]. The Word2DM adapts the Skip-gram objective but instead of vectors, density matrices are being learned. The extraction of density matrices out of quantum circuits is not a trivial procedure and has been marked by the author of [6] as the biggest challenge of their implementation. Density matrices can be constructed

through a process of quantum state tomography, which is a process of reconstructing a quantum state through repeated measurements. A comprehensive overview of this topic can be found in [2]. Another challenge the author faces with this implementation is the computation of the gradient. Unfortunately, the author cannot find a strategy for computing the gradient for real quantum hardware. The experiment is only executed on a simulator, where the gradients can be computed analytically [6].
To summarize the procedure of the experiment: The model is presented with a Word2DM task. To extract the density matrices the circuits are not just simply measured on the computational basis, but instead, quantum state tomography is used to extract them. For evaluation of the model afterwards the similarity of density matrices is calculated. This is achieved by calculating the trace inner product. The trace inner product serves as the matrix equivalent of the vector inner product [6]. For a detailed description of the mathematical process, the reader is referred to section 4.3 in [6]. It holds now, that the trace inner product of similar density matrices approaches 1 and the trace inner product of dissimilar density matrices approaches 0.
For the execution of the experiment, two datasets are used. The first one, named ABC, is a toy dataset with three sentences having three words each. The second one, named Animal, has nine sentences with a vocabulary of 25 words. For both datasets, two iterations of a model are being computed. For the first iteration of the quantum model, one BasicEntangler quantum layer is used. In the second iteration, a second layer of the same kind is used. A BasicEntangler layer consists of a row of R_x gates followed by a row of CNOT gates, which connect all qubits which each other. For each iteration, the model is trained for 10 independent runs with each having 100 epochs.
For the evaluation of the model similarities of density matrices are calculated. A model with one layer, which is trained on the ABC dataset, averages over 10 runs a similarity for neighboring words of 0.71 and 0.08 for distant words. Note that neighboring words shall have high similarity and distant words shall have low similarity. Surprisingly, for 2 layers on the ABC dataset, the model averages, again over 10 runs, a similarity for neighboring words of 0.70 and 0.35 for distant words. One might predict that the model performs better with more layers. For the Animal dataset, the author states, that one layer is not sufficient, so the experiment is only run on two layers. They report an average 0.46 for neighboring words and 0.17 for distant words.
The author claims, that increasing the size of the dataset decreases its performance [6]. While with the Animal dataset, a clear distinction can be made between similar and dissimilar words. This might not hold for even larger datasets. Based on this outcome,

the author suggests using other circuit designs or changing the measure of similarity. For further work, they state optimizing circuit ansatz is an important topic, as well as investigating the measure of similarity for density matrices. As mentioned beforehand, a further topic for research is the calculation for a gradient on quantum hardware, since a quantum gradient could not be found for this experiment [6]. Also, the extraction of the density matrix is yet computationally costly on current NISQ devices.

6.1.4. Functorial language modeling

In [59] the authors describe a DisCoCat experiment, in which they let a model predict a masked word inside a sentence. Beforehand it should be said that this is not a quantum experiment, instead they use the DisCoCat framework and combine it with classical machine learning. The masked language learning approach combined with the DisCoCat framework does not define a language model in the usual sense as one might assume, because the whole DisCoCat framework assumes that text data comes annotated with grammatical structure [59].
The masking is achieved by replacing the encoding of the masked word with an identity encoding. Naturally, the model predicts the missing word in a sentence diagram that has a gap. The following diagram shows the intuition behind this idea. In the sentence *Alice cooks dinner at home* the word *cooks* gets masked:



The authors implement a small proof of concept model for which they use cross entropy loss [22] as their objective function and a combined sum of $l1$ and $l2$ regularization [43]. The optimizer is Adam [30] with a learning rate of 0.05.
Their model is trained on a corpus with 86 sentences [59]. For previous unseen data (23 sentences), the model can predict the missing word inside a sentence with an accuracy

of greater than 75 percent [59].
For future work, the authors try to implement their idea on raw text data and attempting to let the model learn both the probabilistic grammar and the DisCoCat model simultaneously.

6.2. Overview of the word prediction experiment

The experiment will deal with a word prediction task in a quantum machine learning setup. In particular, a masked language learning approach is tested, similar to the one in BERT, but without next sentence prediction. We also simplify the process shown in section 2.2 and only mask one word per sentence. Masked language modeling is chosen, since in a particular sentence the model will get information from both the right and the left context of a masked word, as seen in the definition 5.4.1. We will solely focus on the pure word prediction task, on how can it be achieved and on what does it imply in a quantum context. We will not focus on any downstream task performance.
Every forward pass is pure quantum. For the implementation of the model, the Lambeq package [28] is being used. Quantum circuits will be noiselessly simulated using a local backend. The outcome of these computations would portray the performance of a faultless quantum computer.
The following section will describe the hyperparameter for this experiment. It will be split into classical and quantum hyperparameters. In the end, possible problems will be discussed when dealing with statistical syntax parsers in the DisCoCat framework.

6.2.1. Quantum hyperparameters

Number of qubits: As seen in section 5.4.1 it is possible to choose the amount of qubits, which represents the Hilbert space of the basic types. When working with the English language there are 5 atomic types: *noun*, *noun_phrase*, *sentence*, *prepositional_phrase* *conjunction*, and *punctuation* [28]. The dataset in this experiment only deals with *noun, sentence, prepositional_phrase*. Note that it is possible to assign a different amount of qubits to different types, such as N:2 and S:1, but every occurrence of a type always has the same amount of qubits.

Number of layers: It indicates how many iterations of a chosen ansatz are stacked on top of each other. Note that this hyperparameter increases the number of parameters.

Number of single qubit parameters: This hyperparameter describes how many iterations of $R_z(\theta)R_x(\theta)R_z(\theta)$ gates are being used for boxes, which are represented by exactly one qubit. Less than three single qubit parameters are suboptimal since they would not be able to represent states on the entire Bloch sphere.

Ansatz type: As seen in section 3.5.2 there exist several types of ansätze. Which ansatz is chosen is described with this hyperparameter.

6.2.2. Classical hyperparameters

Epochs: The number of epochs describes how many times the model is trained on the training corpus. For each epoch, the training data is randomly shuffled.

Learning rate: The learning rate defines the pace at which the model is updated in the direction of a minimum. This parameter is part of the optimizer. For a more detailed description of the learning of models compare section 3.5.3.

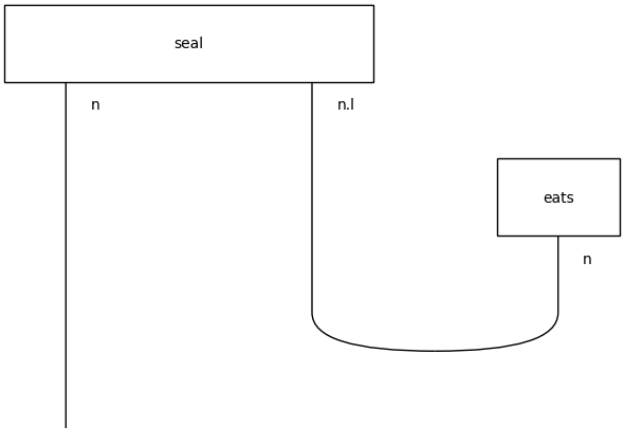
The parameter shift scaling factor: Remember that in quantum machine learning the SPSA [50] algorithm is used, which uses random perturbations to estimate the gradient of the objective function without requiring explicit knowledge of its analytical form. The parameter shift scaling factor indicates how strong the perturbation is. In the mathematical description of SPSA in section 3.5.3 it is referred to as c.

Loss function: The loss function, also known as the objective function or the cost function, is a function that quantifies how well a model performs. In particular, it quantifies the discrepancy between the predicted output of a machine learning model and the desired output. For a complete mathematical description of the loss function in this experiment, the reader is referred to appendix F.

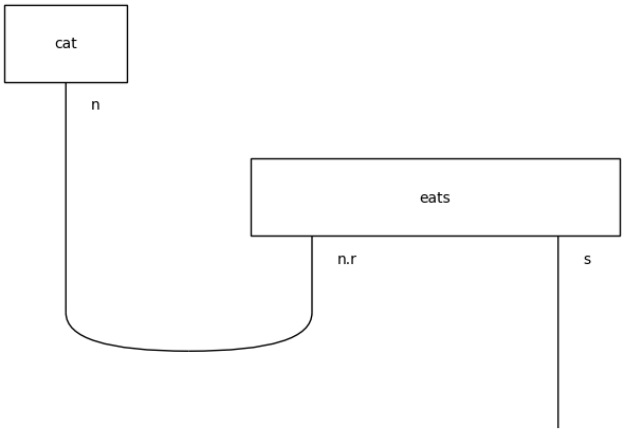
6.2.3. Problems with statistical syntax parsing

In section 5.1 the need for a mapping from every word of a language to a pregroup grammar type is explained. We also need a sentence to be parsed in the pregroup grammar. The Lambeq library [28] suggests using a statistical parser, which is based on a classical large language model. This poses a serious problem for our experiment since

not every sentence of our dataset was recognized as a sentence. Consider the following parsed sentence. It is generated with the BobCat parser [9]:



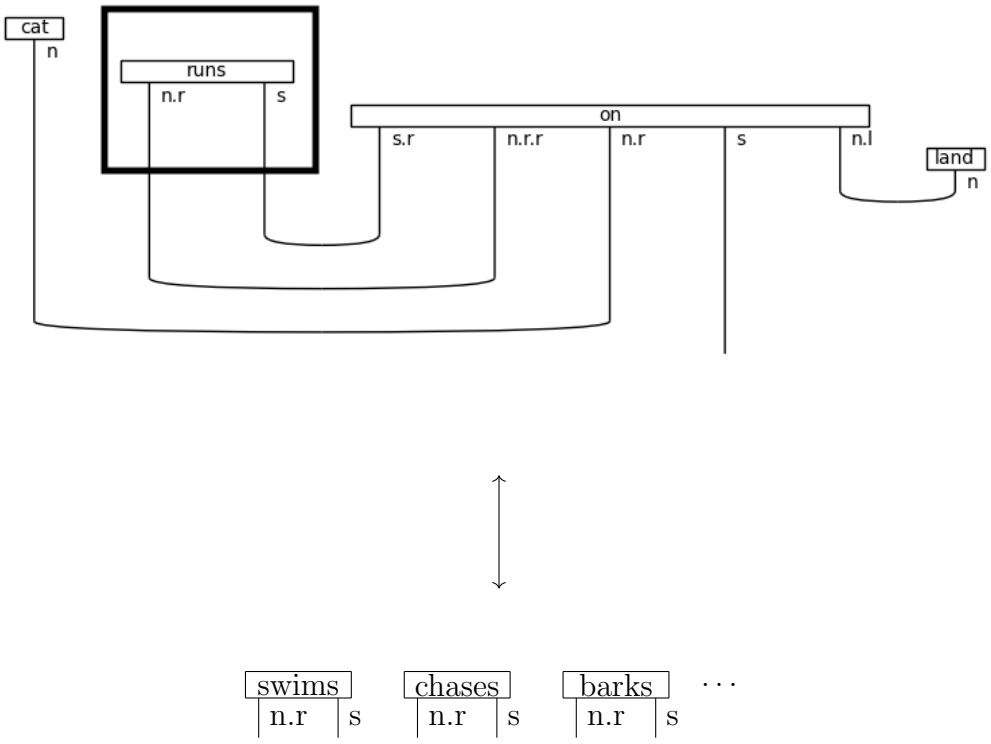
The open wire of this sentence is of type n, therefore, reduces to that type also. Consider the next sentence:



This sentence reduces to type s. In order to make reliable predictions, every sentence needs to be reduced to the s type. This fact comes apparent when the number of qubits gets scaled. When we have different dimensions of the output wires for different sentences, we also have different model output sizes for different sentences. This would be unacceptable. To mitigate this problem, we wrote our own grammar and parser for this dataset. With a rule-based parser, the problem is solved. Of course, this approach is not meant for scaling.

6.3. Word prediction as binary classification

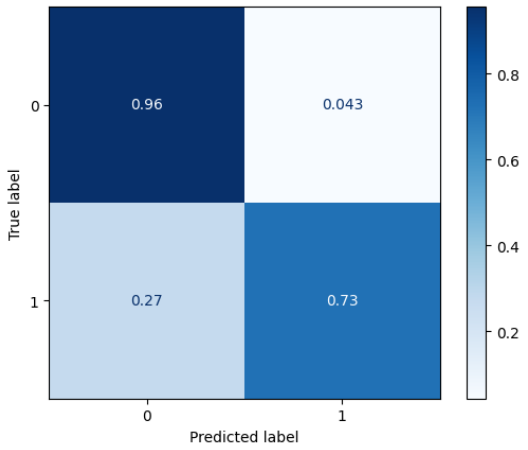
In general, word prediction for a masked token refers to a multiclass classification task (presuming you have a vocabulary size greater than two), since the model has to decide between multiple words. The prediction usually has the size of the vocabulary. Since successful past works in this area [40] [34] deal with binary classification it is of interest to rethink masked language modeling for quantum natural language modeling in terms of a binary classification.
Multiclass classification can be transformed into binary classification with a one-versus-one task. This means that the masked word gets replaced by every other word in the vocabulary. The correct sentence is labeled positive, the other sentences get the negative label. Now the problem comes, how do we deal with non-grammatical sentences that will inevitably come up? The DisCoCat framework can not deal with ungrammatical sentences. Luckily, there exists a solution to that problem. If we parse the grammatically correct sentence, we can see how many output wires the masked word has. We then replace the masked word with words that have the same amount of output wires and therefore create negative examples. We can guarantee the grammatical correctness of both negative and positive examples. The strategy is illustrated with the following picture, where *runs* is the masked token and gets replaced by *swims*, *chases* or *barks*:



It is now quite obvious to see, that with this method the negative training sentences, i.e. ones with the label false, heavily outweigh the positive training examples. Empirical analysis with that strategy revealed that negative examples outweigh positive ones by a factor of around 8. This leads to a big problem with the model, since it became immensely biased towards negative labels and only predicted negative labels. This leads to a very poor F1-score. To mitigate this problem, we balance the dataset with positive examples, so that negative and positive labels are roughly equally represented in the corpus. This one-versus-one strategy blows up the training corpus and produces a training corpus with 1346 training sentences and 358 test sentences.

6.3.1. Evaluation

To properly evaluate this model, accuracy, and F1-scores on unseen test sentences are calculated. The best model we could compute leads to an accuracy of 82.96%. The following two images show a more detailed analysis of this model. On the left, there is the normalized confusion matrix and on the right, we show the classification scores including F1-scores:



	precision	recall	f1-score	support
0	0.74	0.96	0.83	161
1	0.95	0.73	0.82	197
accuracy			0.83	358
macro avg	0.85	0.84	0.83	358
weighted avg	0.86	0.83	0.83	358

Figure 6.3.: Normalized confusion matrix and F1-scores for the binary approach

If the reader is not familiar with those measures we refer to [48] and [60]. The binary method can achieve similar or slightly better results than the approach taken in [59]. However, comparing the accuracy of our binary classifier with that of [59] is hard due to inherent differences in their nature. Due to the immense computing cost of simulating many quantum circuits, we were limited in the scaling of the model. There is still the opportunity to scale even more with the number of qubits, the number of layers and the number of parameters. To properly show the development of such models we also document the hyperparameter search in table 6.1. We focus on optimizing quantum hyperparameters and the loss function since there is extremely little literature on that topic, especially regarding the DisCoCat framework.

The following table 6.1 summarizes all our results regarding the binary classification task. All the models were trained on 120 epochs and for each epoch evaluated on the train and test set. The optimizer was the SPSA algorithm since the standard parameter shift rule would be too expensive. For all models, we used a learning rate of 0.2 and a parameter shift scaling factor of 0.06. q_n describes the number of qubits for the noun type, q_p the number of qubits for the prepositional phrase type, and q_s the number of qubits for the sentence type. The number of qubits for the sentence type in the binary case is always 1. For a more detailed explanation of how the sentence wires are connected to the task, please confer section 6.4.1. n_{layers} indicates the number of layers for multi qubit boxes. n_s refers to the single qubit parameters. Note that if both the noun type and the prepositional phrase type get mapped to more than one qubit, there are no more boxes with single qubit parameters. *bce* and *hinge* refer to the binary cross entropy loss and hinge loss respectively. For a detailed description of the loss function please consider appendix F. Every hyperparameter configuration is run five times. For the first and second evaluation metrics, the arithmetic mean of the highest accuracy on test and train data are calculated respectively.

ID	model configuration	ø highest acc on test data	ø highest acc on train data	model parameters
1	$q_n = 1; q_p = 1; q_s = 1; n_{layers} = 1; n_s = 3$ $loss = bce; ansatz = IQP$	0.7111	0.7587	76
2	$q_n = 1; q_p = 1; q_s = 1; n_{layers} = 4; n_s = 8$ $loss = bce; ansatz = IQP$	0.73296	0.79748	284
3	$q_n = 1; q_p = 1; q_s = 1; n_{layers} = 4; n_s = 8$ $loss = hinge; ansatz = IQP$	0.781	0.8561	284
4	$q_n = 1; q_p = 1; q_s = 1; n_{layers} = 7; n_s = 13$ $loss = hinge; ansatz = IQP$	0.77318	0.85588	420
5	$q_n = 1; q_p = 1; q_s = 1; n_{layers} = 10; n_s = 18$ $loss = bce; ansatz = IQP$	0.70282	0.69196	592
6	$q_n = 2; q_p = 2; q_s = 1; n_{layers} = 4$ $loss = hinge; ansatz = IQP$	0.7631	0.8514	312
7	$q_n = 2; q_p = 2; q_s = 1; n_{layers} = 7$ $loss = hinge; ansatz = IQP$	**0.7989**	**0.9301**	546
8	$q_n = 1; q_p = 1; q_s = 1; n_{layers} = 4; n_s = 8$ $loss = hinge; ansatz = SIM14$	0.77264	0.80774	944
9	$q_n = 1; q_p = 1; q_s = 1; n_{layers} = 4; n_s = 8$ $loss = hinge; ansatz = SIM15$	0.75476	0.80742	528
10	$q_n = 1; q_p = 1; q_s = 1; n_{layers} = 4; n_s = 8$ $loss = hinge; ansatz = StrongEnt$	0.78266	0.829	736

Table 6.1.: Results of the binary approach to masked language modeling

The best run was achieved with configuration seven. It had on average the best test-scores and training-scores. The model presented in the beginning of this section also has the hyperparameters from configuration seven. This table is further discussed in section 6.3.2.

The following figure shows the training process of a binary classification model with hyperparameter-configuration 7 from table 6.1 above. It shows the convergence of the cost function and the evolution of the accuracy on the training and test set.

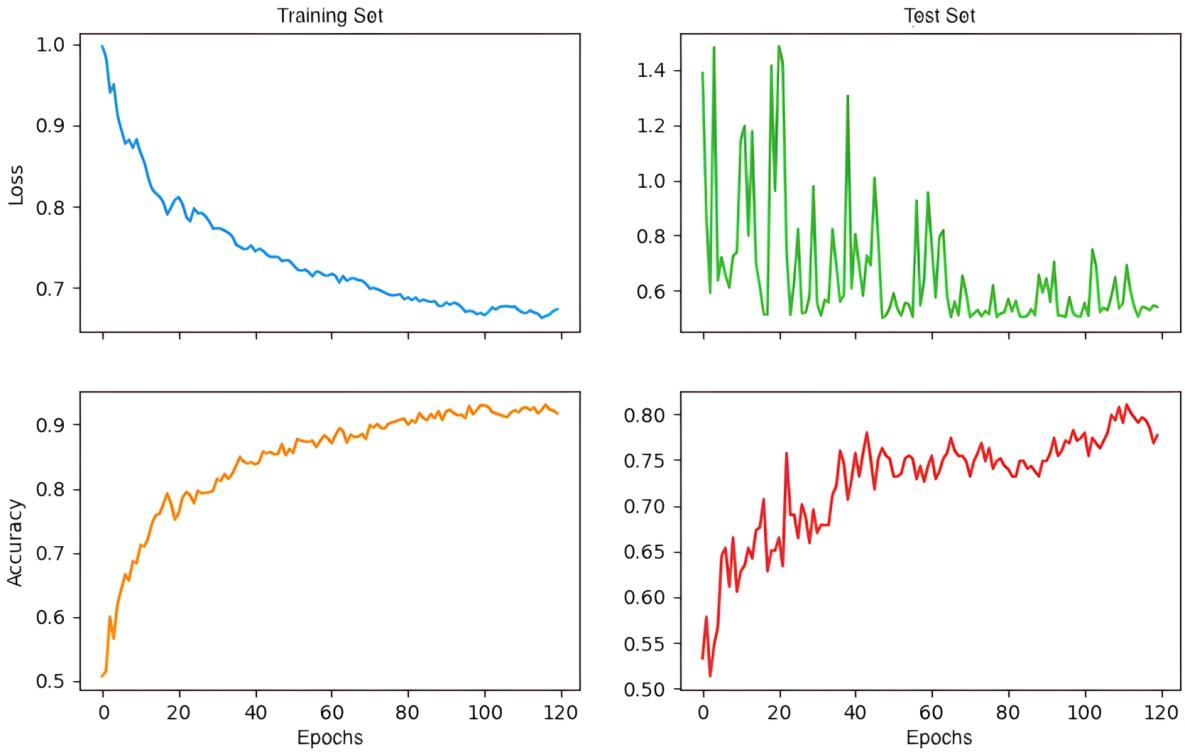


Figure 6.4.: Visualisation of the loss and the accuracy over the course of the training in the binary case

6.3.2. Discussion

We can report that binary classification yields good results. From the confusion matrix in 6.3 we can observe diagonality, which intuitively indicates good performance. But it is also apparent, that the model can classify negative examples slightly better than positive ones. The obtained F1-score of 0.83 reflects the model's strong ability to strike a balance between precision and recall.

From table 6.1 it is immediately apparent, that scaling the model size directly increases model performance. This is of course obvious from a classical machine learning perspective. But it also reaffirms the taken quantum machine learning approach. Comparing configurations 2 and 3, we see that our model performs better with hinge loss than with binary cross entropy. Even when scaling parameters, like in configuration 5, binary cross entropy performs arguably worse in every measurement.

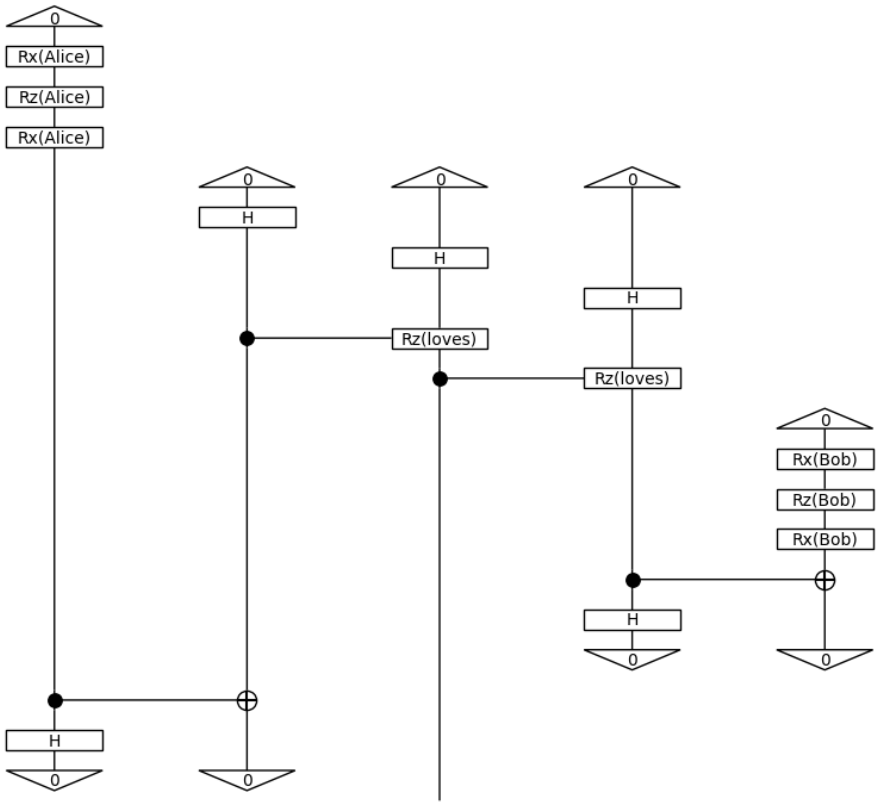
Configurations 8, 9, 10 show the impact of different ansatz types. SIM 14, SIM 15 and Strongly Entangling ansatz perform very similarly to each other. Since we only made observation from five runs each we cannot make definitive answers on whether the ansatz type has strong correlations to model performance. Considering this task, it seems that the ansatz types perform very similarly to each other. But what is very apparent is difference in model size with the ansätze. Comparing configuration 3 with $8, 9, 10$ we see that the IQP ansatz has the least amount of parameters while performing similarly (or even slightly better) to every other ansatz type. Especially the SIM 14 ansatz has around 3.3 times more parameters than the IQP ansatz but no performance improvements. The model size has a huge impact on computing resources. For simulating circuits, the time it takes to train a model with the IQP ansatz is significantly shorter than with the SIM 14 or Strongly Entangling ansatz. The size difference becomes an even bigger impact when letting those models run on NISQ devices. Every additional gate introduces additional noise. Each gate operation introduces the possibility of errors due to imperfections in the gate implementation or interactions with the noisy environment. As the circuit becomes more complex with more gates, the accumulation of errors can degrade the overall accuracy of the computation. It is conclusive to say that the IQP ansatz is a good choice for binary classification tasks in the NISQ era and for simulating circuits.
The best models were achieved with configuration 7. It is still unsure if it is possible to get even more performance out of this limited dataset. Further tests were extremely hard, since the scale of the experiment (in terms of circuit depth, circuit length, and number of circuits) made it extremely hard to simulate, due to RAM issues.
The major downside of the binary classification approach to this pretraining phase is the blowup of the dataset. It has been already said that the dataset went from 86 training sentences to 1346 sentences. This approach of binary classification would lead to extremely large datasets when working with real-world data. To avoid this blowup, maybe it is smart to limit the number of negative examples we allow per training sample. But this topic is still up to further research and discussion.
From figure 6.4 we can see that within the training set, the cost function converges relatively smoothly. In addition, the accuracy rises well for both training and test data. The loss on the test set however seems really volatile. Volatility can be attributed to the fact that we have a high learning rate and few test sentences. We made the choice for a high learning rate since, as of right now, the SPSA optimizer does not feature an adaptive learning rate. The training phase would take significantly longer with a smaller learning rate.

6.4. Word prediction as multiclass classification

As explained above, the binary classification has very good results on one side but has very big problems with scalability. Also, one can argue that word prediction is innately a multiclass problem. These two reasons lead to a development of a multiclass classification model within a quantum machine learning setup. The first challenge for this task is to get the model to predict more labels. The second challenge is finding an appropriate masking approach. The third obstacle is to keep the task complexity as low as possible. In the binary task, it is possible to avoid the masking entirely since we simply replaced the word by other words. The masking strategy is not a trivial task, since it is not possible to do a simple identity masking for quantum circuits. More on that topic can be found in section 6.4.2.

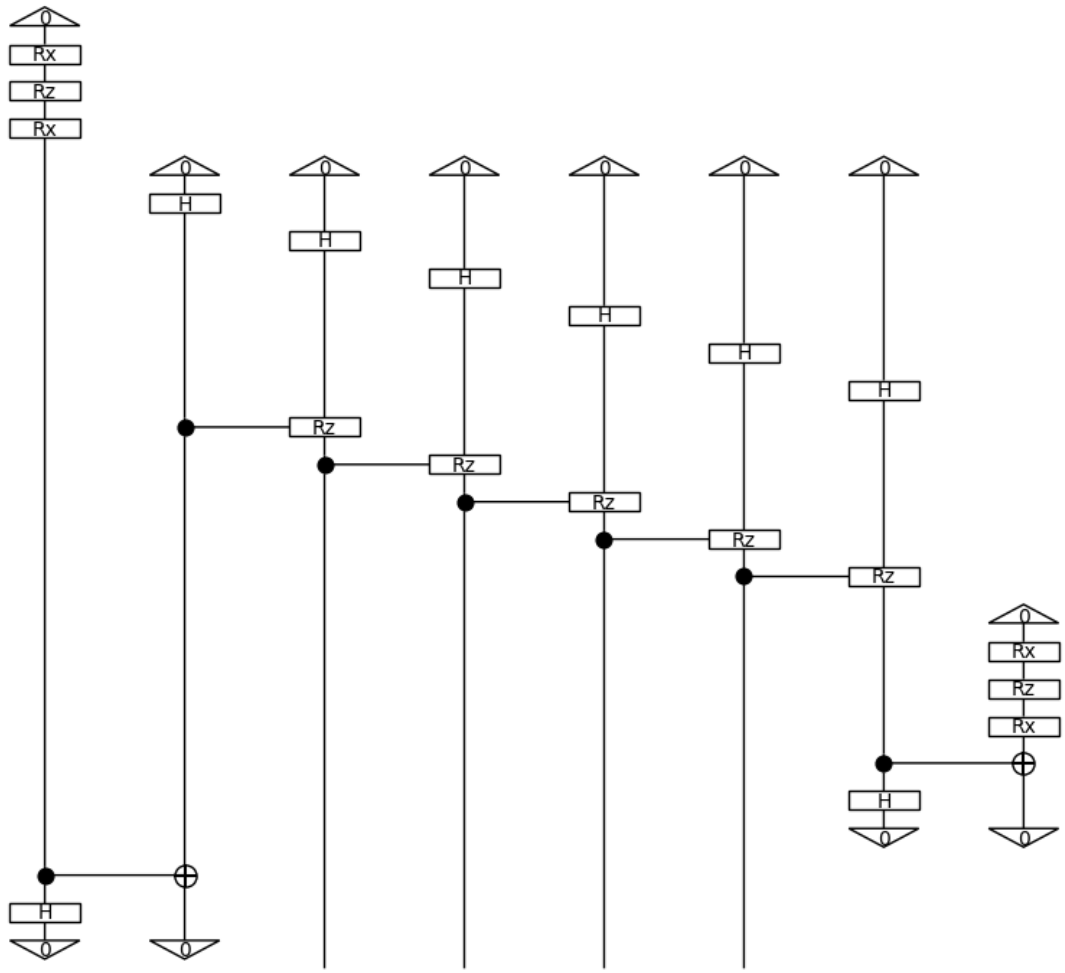
6.4.1. Multiclass classification in DisCoCat

As already mentioned in section 5.4.4, the sentence wire directly refers to the meaning of the sentence. For classification tasks, this meaning is repurposed as the output vector of our model. For binary classification, that means the model output corresponds to the truth value of the sentence so that the model parameters are learned accordingly. The model learns to predict the correct truth value, aka meaning, to every sentence. Consider the circuit on the next page for the often-used example *Alice loves Bob* (IQP ansatz is being used):



The output wire gets measured without any postselection on it, and it corresponds to our model output. The output state of one qubit is in $\mathbb{C}^2$. The output state of two qubits is in $\mathbb{C}^4$. The output state scales with 2^n, where n is the number of qubits. Note that when working with circuits practically, we norm and square the complex values, such that they represent a probability distribution. For more information regarding qubits, we refer to chapter 3.

Now consider the circuit, also on the next page, with $S : 4$, which has an output size of 16:



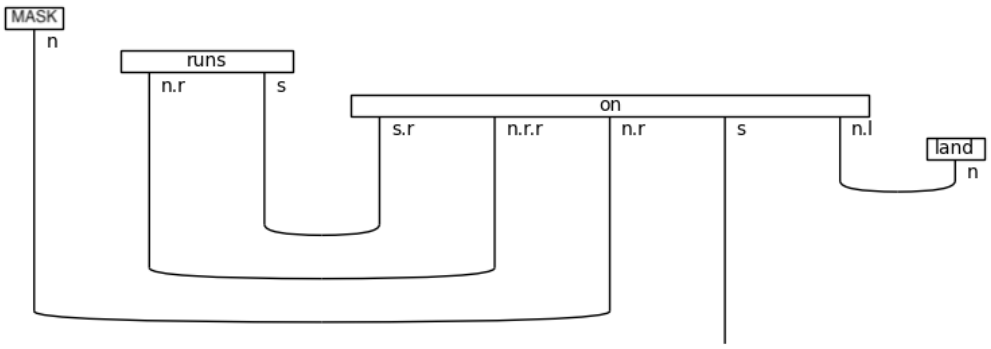
This proves that we can generate an arbitrary multiclass classification by scaling the qubits of the sentence type. Note that due to visibility reasons, no parameters of the rotational gates are displayed.
In the multiclass classification approach, on the other hand, the model output does not correspond to the truth value of the sentence. It corresponds to the one-hot vector for the masked token. For example, if the vocabulary is of size 5, and we mask the second word of it in a sentence, the output should be ideally $[0, 1, 0, 0, 0]$.

6.4.2. Masking strategy

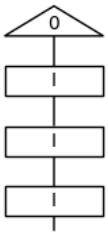
The general idea for masking a word is that we conceal the information for that word, and we let the model predict what word is missing. In this section, we will talk about the different methods that were tried to achieve masking in a quantum computing environment.

Identity masking

In [59] the masking is done by replacing the masked word with the identity matrix, therefore creating a 'hole'. The goal is to predict the missing word of a sentence with a hole. This identity masking was used for the classical approach [59], but the picture is more complex when working with quantum circuits. We will first show examples, where identity masking works with quantum circuits, then explain situations where it doesn't work. Consider the sentence *cat runs on land*, where *cat* gets masked with identity masking:



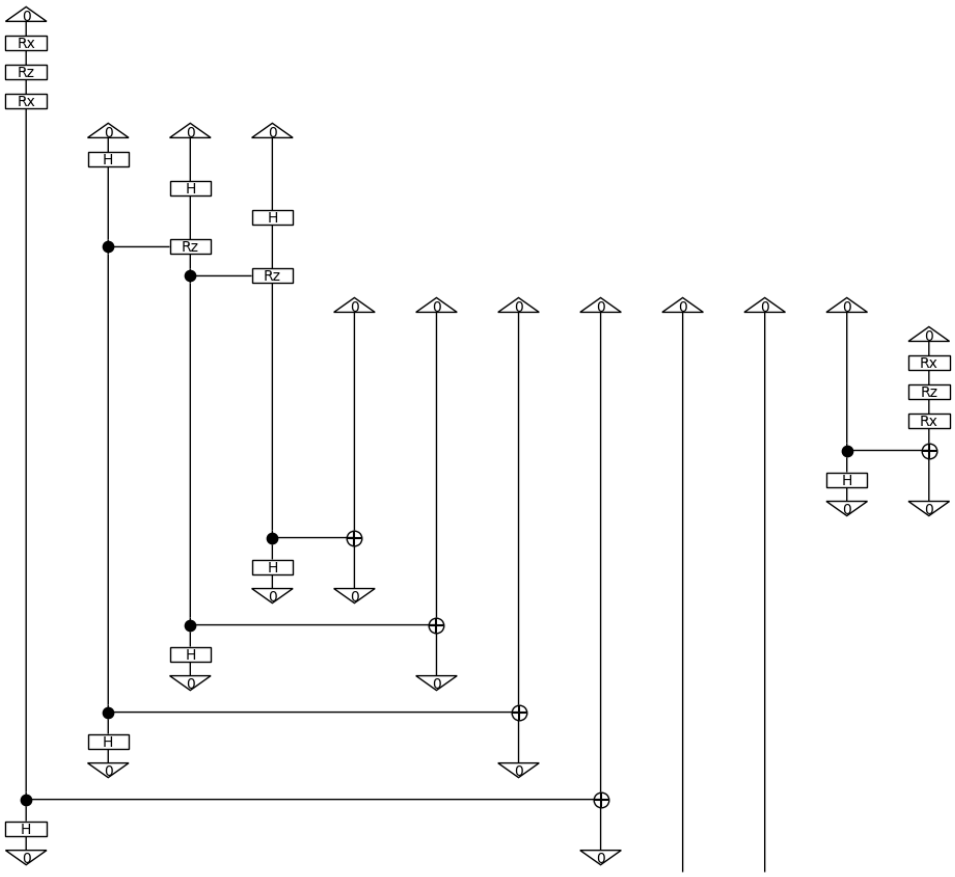
In a quantum circuit, the box for the masked word *cat* looks like this:



Note that due to efficiency reasons, the actual identity gates can be left out. An identity gate has per definition no effects on qubits, it holds:

$$I^{\otimes n} * |\psi^n\rangle = |\psi^n\rangle$$

The complete circuit, where *cat* is masked, with $N : 1, S : 2, P : 1$, one layer of the IQP ansatz and three single qubit parameters can be found in appendix C.1.
This circuit is trainable. That means that we can train the rotational gates in this circuit in a way, that the output wires can represent any of the four possible label combinations: $[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]$. Identity masking also works when we mask the word *runs* or *land*. But the bottom line here is, that identity masking does not work when we mask a word, that contains the output wires.
Consider the circuit where the word *on* is masked:

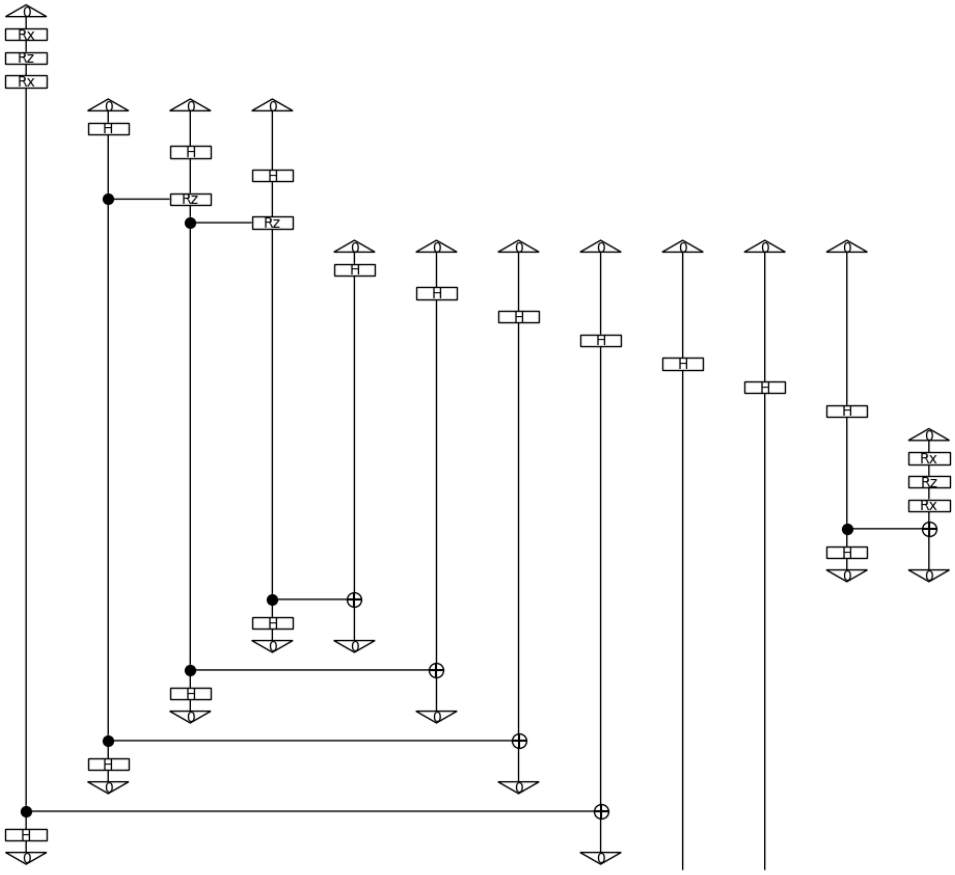


Even though we have postselection on $q_0 - q_7$ and $q_{10} - q_{11}$ this cannot influence the measurements on q_8 and q_9. Every single time, q_8 and q_9 are in state $|00\rangle$. Whatever the impact of postselection might be, it will not impact the probability distribution of the measurement. The non-masked words cannot influence q_8 or q_9.

This statement can be empirically verified. In an isolated way, we tried to train this exact circuit, where *on* is masked, to predict the label vector $[0, 0, 0, 1]$. First, in every DisCoCat model, the parameters are initialized randomly. With random parameters and no training, the model predicted the vector $[1, 0, 0, 0]$. In a quantum machine learning setup, this should be already a hint that something is not working right. The training was not possible and after 500 epochs the output vector was still $[1, 0, 0, 0]$. We could not train this circuit using identity masking.
In comparison, we tried to train the circuit, where *runs* is masked. With random parameters and no training the model had the output vector $[0.3436, 0.4553, 0.1174, 0.0846]$. Note that, since the model-output are the probabilities of each state, they add up to 1. The model quickly converged after around 20 epochs, and after the 250 epochs produced an output vector of $[0.0, 0.0, 0.0, 0.999871]$. This isolated training was successful. We achieved similar results for *cat* and *land*, thus concluding that identity masking works when we mask boxes that have no output wire but does not work when masking words that have an output wire.

Hadamard masking

A very similar approach to identity masking is Hadamard masking. Instead of applying identity gates (or no gates in the example above), we applied a layer of Hadamard gates. A masked circuit looks like this:

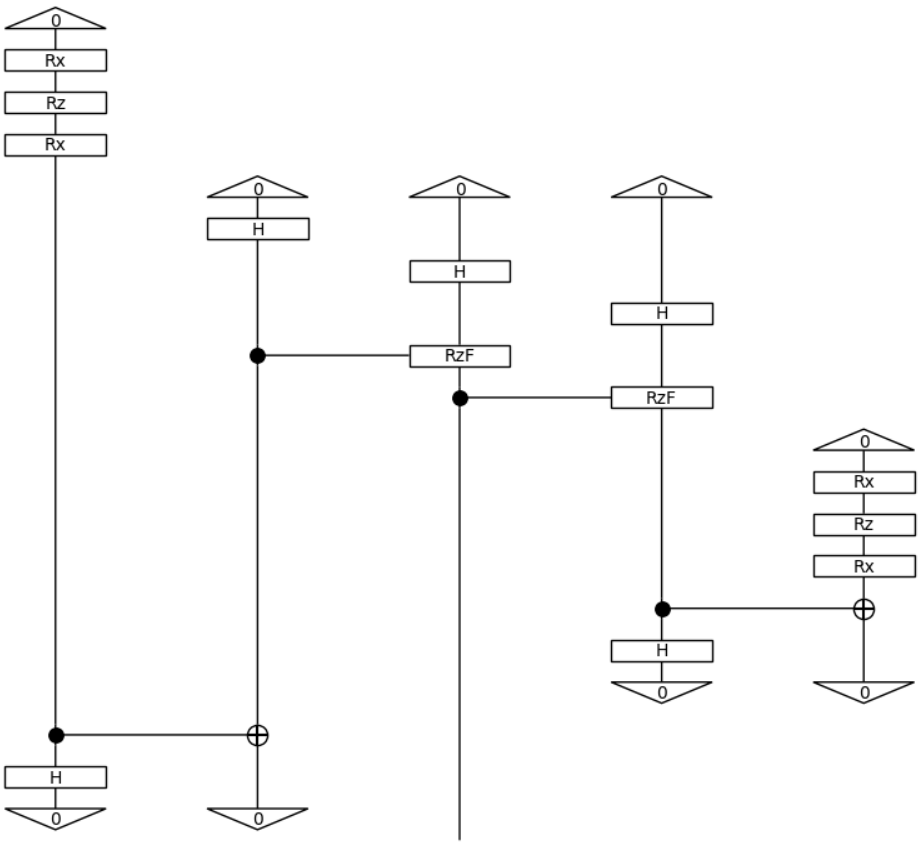


This sort of masking has the same problems as the identity masking approach. One can train circuits, where boxes with no output wires are masked but cannot train circuits where boxes are masked, that have output wires. This statement can also be empirically verified.

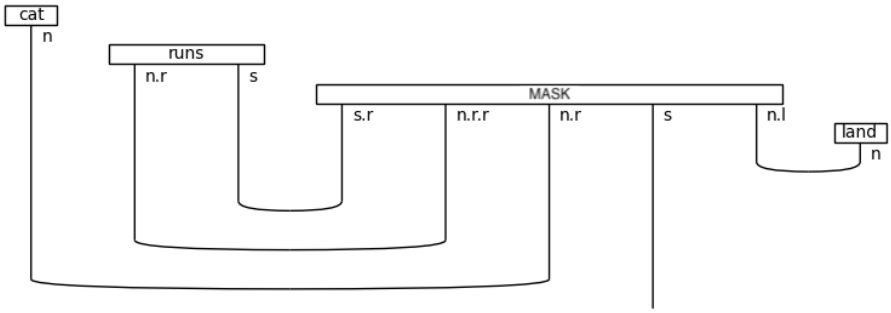
Fixed parameter masking

The final masking approach explored in this thesis is a masking approach where we fix the rotation angles of the masked word and therefore only train the parameters of the non-masked works. We do this by keeping the ansatz architecture for the masked word, applying random rotation angles on the gates, but fixing them so that they cannot be impacted by training. For masking a single qubit box, we keep the $R_z(\theta)R_x(\theta)R_z(\theta)$ structure as well, and apply random but fixed parameters. Therefore, we do not create

a hole for the masked token, but apply a fixed blur over the masked token and let the model predict the blurred word. When we mask *loves* in the example sentence *Alice loves Bob* the circuit looks the following way:



Note that all normal parameter-gates are depicted as standard R_x and R_y and the fixed parameter gates are depicted as R_zF and R_xF.
We would like to show empirically that this ansatz is a possible solution to the problems, which the other masking approaches have. In an isolated experiment, we try to learn a model with the sentence *cat runs on land*, where *on* is masked. It is immediately apparent, that *on* is a multi qubit box, where the output wires are located. In this situation, all other masking approaches failed:



The circuit for this sentence is found in appendix C.2. Before training, the model associates an output vector of $[0.19, 0.39, 0.16, 0.25]$ to the training sentence. We label the sentence with the vector $[1, 0, 0, 0]$. Afterwards, we train the model and the model is able to converge and predict the vector $[0.98, 0.00, 0.12, 0.00]$. This method worked for the other label vectors equally well:

training label	initial vector	label after training
$[1, 0, 0, 0]$	$[0.19, 0.39, 0.16, 0.25]$	$[0.98, 0.00, 0.12, 0.00]$
$[0, 1, 0, 0]$	$[0.27, 0.01, 0.51, 0.19]$	$[0.01, 0.97, 0.01, 0.01]$
$[0, 0, 1, 0]$	$[0.10, 0.03, 0.22, 0.63]$	$[0.02, 0.01, 0.95, 0.02]$
$[0, 0, 0, 1]$	$[0.47, 0.47, 0.02, 0.02]$	$[0.02, 0.01, 0.00, 0.97]$

Table 6.2.: Training with the fixed parameter masking approach

Note that this masking approach successfully made it possible to train masked sentences, where the mask and the output wires overlap on the same box.
In general, the downside of a multiclass classification approach with masking is scaling. In our tests, it comes apparent that the model needs to be scaled drastically, in order to learn predictions in terms of a vector with more than four dimensions. Therefore, it is of great interest to have a label vector with as few dimensions as possible.
For a mathematical explanation of the fixed parameter masking with the connection to the cups and the postselection, the reader is referred to appendix A. In short, by essentially creating a random probability distribution and connecting the qubits with each other, we can influence the output wires with the help of cups and postselection in such a way, that training is possible.

6.4.3. A method for reducing the number qubits

It seems obvious, that when predicting a masked token, the size of the output vector corresponds to the size of the vocabulary. In the experiment proposed here, that would mean an output vector the size of 1×28 to represent all the vocabulary. In a quantum circuit context, that would mean, that we need to have at least five output qubits, since $2^4 < 28$. Since $2^5 = 32$, we would need to always fill the surplus four positions with zeros. Due to computing limitations, classically and quantum, it is of great interest to save on qubits wherever possible.
To save on qubits, we organize the vocabulary. Firstly, we parse all the sentences and remember for every word, how many output wires its box has. With that information, we generate an organized vocabulary with groups, where each group corresponds to a different sizes of boxes. Each group has a list of words:

box size	words
1	[dog, chicken, grain, ...]
2	[runs, swims, barks, ...]
3	[eats, chases, bites, ...]
5	[on, after, in,...]

Table 6.3.: A example of an organized vocabulary

Note that with the toy dataset here, we do not generate boxes with four output wires. Now we set the potential output vector length to the size of the biggest group in the organized vocabulary. This vector needs to be represented by the power of two. In this toy case, the group for the box size of 1 has 14 words dedicated to it and is the biggest group. The output vector would be of size 1×14. In quantum computing terms, this would be represented by 4 qubits, since $\lceil \log_2(14) \rceil = 4$. Recall that initially, without employing this strategy, we would have required five qubits. However, thanks to this approach, we are able to save one qubit, making the entire process more efficient.

Words are then represented with a 1 in the label vector at the index in the according group.

$$word_i = [x_1, x_2, \ldots, x_n] \begin{cases} x_j = 1, & j \neq i \\ x_j = 0, & j = i \end{cases}$$

$$i = \text{index of word in the according group}$$

$$j = \text{index variable of the label vector}$$

$$k = \lceil \log_2(\text{size of the biggest group}) \rceil$$

$$n = 2^k$$

For example, the biggest group has 5 members, therefore is the output vector of size 1×8. If we mask the second word of the first group, it is represented by the vector $[0, 1, 0, 0, 0, 0, 0, 0]$. If we mask the fifth word of the third group, the vector is of form $[0, 0, 0, 0, 1, 0, 0, 0]$. To retrieve a predicted word, which means to obtain a word from a vector, the model looks at the number of wires the masked box has and recover the word in the according box.

Advantages: The benefit of this approach is to save on qubits. But this approach also reduces the complexity of the task. The more output cases the model has to differentiate between the harder it is for the model.
Limitations: It is yet unclear to see if this method is still useful when working with large-scale data. The possible amount of states a qubit can carry scales exponentially with 2^n, where n is the number of qubits. When organizing a large-scale vocabulary it might not be the case that the biggest group and the overall number of words can be represented with two different amounts of qubits. It is yet also unclear to see if this strategy impacts downstream task performance.

6.4.4. Evaluation

The second part of the experiment also features the evaluation of the multiclass classification approach. Note, that this time, there is no blowup of training and test sentences due to balancing like in the binary classification approach. Therefore, these models are trained on 86 training sentences and evaluated on 23 test sentences. For these models, we also used the method for reducing the number of qubits, as explained in section 6.4.3. The best model we could compute with the resources at hand led to an accuracy of only

34.78%. For the sake of completeness we also show an unnormalized confusion matrix and the F1-scores. Due to having only 23 test sentences, the confusion matrix is very sparse:

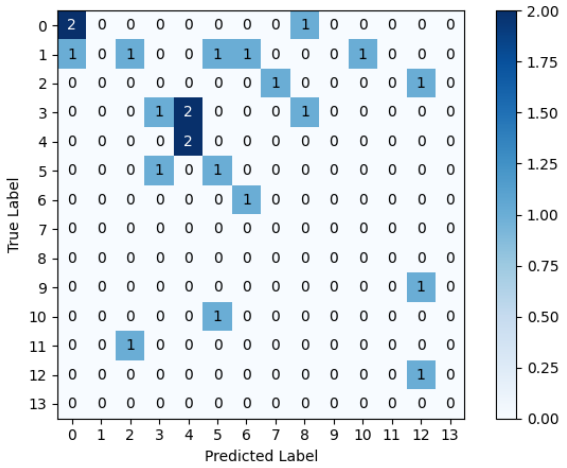


Figure 6.5.: Unnormalized confusion matrix for the multiclass approach

	precision	recall	f1-score	support
0	0.67	0.67	0.67	3
1	0.00	0.00	0.00	5
2	0.00	0.00	0.00	2
3	0.50	0.25	0.33	4
4	0.50	1.00	0.67	2
5	0.33	0.50	0.40	2
6	0.50	1.00	0.67	1
7	0.00	0.00	0.00	0
8	0.00	0.00	0.00	0
9	0.00	0.00	0.00	1
10	0.00	0.00	0.00	1
11	0.00	0.00	0.00	1
12	0.33	1.00	0.50	1
13	0.00	0.00	0.00	0
accuracy			0.35	23
macro avg	0.22	0.34	0.25	23
weighted avg	0.28	0.35	0.29	23

Figure 6.6.: F1-scores and accuracy for the multiclass approach

This method performs worse than the binary one or the classical approach from [59]. In section 6.4.5 we will discuss the possible reason for that poor performance. We continue with the hyperparameter part:

All the models were trained on 200 epochs and for each epoch they were evaluated on the train and test set. The optimizer was again SPSA. For all models, we use a learning

rate of 0.2 and a parameter shift scaling factor of 0.06. We use cross entropy as a loss function, see appendix F. For a detailed explanation of all parameters used in the following table, the reader is referred to section 6.3.1 and 6.2.

ID	model configuration	ø highest acc on test data	ø highest acc on train data	model parameters
1	$q_n = 1; q_p = 1; q_s = 4; n_{layers} = 1; n_s = 3$ $loss = ce; ansatz = IQP$	0.1652	0.2256	130
2	$q_n = 1; q_p = 1; q_s = 4; n_{layers} = 4; n_s = 8$ $loss = ce; ansatz = IQP$	0.26086	0.479	464
3	$q_n = 1; q_p = 1; q_s = 4; n_{layers} = 7; n_s = 13$ $loss = ce; ansatz = IQP$	0.26088	0.58374	798
4	$q_n = 1; q_p = 1; q_s = 4; n_{layers} = 10; n_s = 18$ $loss = ce; ansatz = IQP$	0.2695	0.5744	1132
5	$q_n = 1; q_p = 1; q_s = 4; n_{layers} = 13; n_s = 23$ $loss = ce; ansatz = IQP$	0.2869	0.6303	1466
6	$q_n = 1; q_p = 1; q_s = 4; n_{layers} = 16; n_s = 28$ $loss = ce; ansatz = IQP$	0.2608	0.5791	1800
7	$q_n = 2; q_p = 2; q_s = 4; n_{layers} = 13$ $loss = ce; ansatz = IQP$	0.2522	0.67442	1677
8	$q_n = 2; q_p = 2; q_s = 4; n_{layers} = 16$ $loss = ce; ansatz = IQP$	**0.29562**	**0.679**	2064

Table 6.4.: Results for the multiclass classification approach

The following figure shows the training process of a multiclass classification model with hyperparameter-configuration 8 from table 6.4 above. It shows the convergence of the cost function and the evolution of the accuracy on the training and test set.

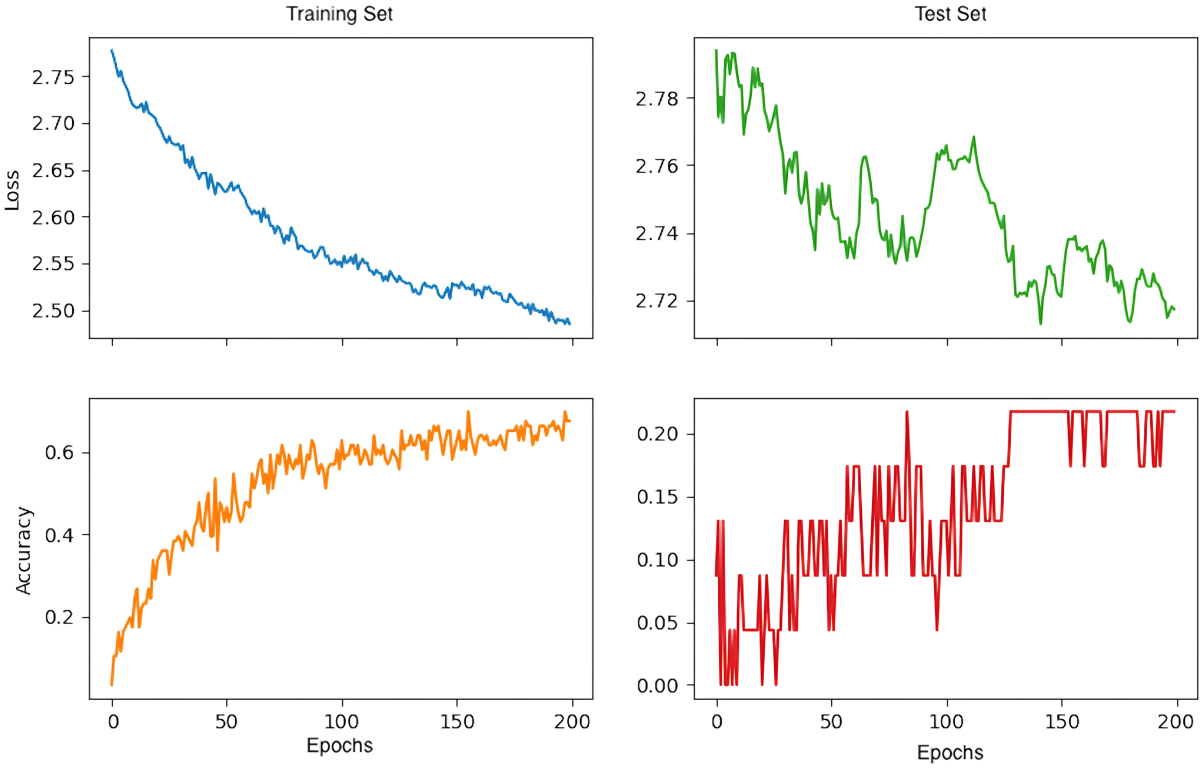


Figure 6.7.: Visualization of the loss and the accuracy over the course of the training in the multiclass case

6.4.5. Discussion

The results of the multiclass classification approach are worse than the ones in the binary classification approach. This is reflected in the accuracy on the training data as well as in the accuracy on the test data. In the confusion matrix 6.5 we cannot observe a diagonal line. The F1-score does also reflect the poor performance. We want to give some possible explanations as, why the multiclass classification approach performs worse.
First, the task for a machine learning model becomes harder when it needs to differentiate between more labels. Remember that for the multiclass classification approach, it has to differentiate between 14 real classes (remember the output vector is still of size 1×16) in comparison to two in the binary approach. With more labels, the general output space increases and the model has to do more fine-grained distinction between the labels. With more labels, the available training data for each individual label may become sparser too. This means that the model may have fewer examples to learn from

for each class, making performance worse.
When facing those problems, one would suggest increasing model complexity to increase the performance. Increasing model complexity comes with problems in the current era of quantum computing. We see in table 6.4 that increasing the circuit width and depth, i.e. qubits and layers, does increase the model performance. But that comes with an immense cost, especially when increasing the number of qubits. The computational complexity of simulating a quantum circuit grows exponentially with the number of qubits in the system. We tried to increase the number of qubits of the quantum circuits, in particular, we wanted to have the following hyperparameters: $q_n = 3;\ q_p = 3;\ n_{layers} = 13$. Having a model with such configuration was not possible, due to immense memory requirements. The reader is referred to appendix E for a more technical description of the problem at hand. Especially for a multiclass classification approach scaling of quantum circuits would be beneficial. A search for a better ansatz type or loss function for this multiclass classification task is beyond the scope of this thesis.
The strategy for reducing the number of qubits shows a really good performance boost. By using this method, we do not only save qubits overall, but we also reduce the number of labels, the model must differentiate between. We train a few models with configurations showcased in table 6.4, but without the qubit saving method. Table 6.5 shows the comparable results:

ID	model configuration	ø highest acc on test data	ø highest acc on train data	model parameters
4	$q_n = 1; q_p = 1; q_s = 5; n_{layers} = 10; n_s = 18$ $loss = ce; ansatz = IQP$	0.15652	0.43488	1322
5	$q_n = 1; q_p = 1; q_s = 5; n_{layers} = 13; n_s = 23$ $loss = ce; ansatz = IQP$	0.1913	0.47208	1713
6	$q_n = 1; q_p = 1; q_s = 5; n_{layers} = 16; n_s = 28$ $loss = ce; ansatz = IQP$	0.1565	0.5139	2104

Table 6.5.: Results for multiclass classification approach without the qubit reduction trick

This strategy shows a performance boost of around 10% on training and test data and a reduction in model parameters as well. A reduction in model parameters is helpful since it decreases the size of the model and makes it computationally easier to work with. Overall, this strategy shows successful results and has the potential to be applied to different tasks.

Based on the observations from figure 6.7, it becomes evident that the multiclass classification approach exhibits greater volatility compared to the binary one. This volatility is particularly notable in the test set, where we observe a higher frequency of fluctuating and irregular patterns. There are two primary factors contributing to these findings. Firstly, the limited number of training and test sentences plays a significant role, which is particularly evident in the straight lines depicted in the bottom right graphic, which displays the accuracy on test data. Secondly, the occurrence of such behavior can also be attributed to a high learning rate, which is further elaborated upon in section 6.3.2. Additionally, the results reveal a high loss in both the training and test set, indicating a deficiency in the model's complexity as the likely cause.

The next aspect we want to discuss is the high amount of epochs used for training these models. As it is the case right now, the optimizer for QNLP experiments is the SPSA algorithm. This algorithm does not yet feature an adaptive learning rate, like modern optimizers such as Adam [30] or AdamW [35]. This can explain slow convergence and therefore the need for a high number of epochs.

Throughout our research in quantum natural language processing, we have successfully computed numerous models without encountering any instances of a barren plateau. This observation strongly suggests that DisCoCat exhibits remarkable robustness against barren plateaus.

7. Conclusion

In this section, we want to give a short summary of our thesis, show the limitations of our contributions, give an outlook, and provide possible further research questions.

7.1. Summary

This thesis deals with the implementation of word prediction in QNLP at a fundamental level. In particular, it explores and analyzes how to embed two possible masked language modeling approaches into quantum natural language processing. Masked language modeling shows promising results as a binary classification task and weak results as a multiclass classification task. Here is a short summary of our findings:

Binary approach: This method transforms the prediction of a masked token, which is inherently a multiclass classification task if there are more than two words in the vocabulary, into a binary classification task. It is done by a one-versus-one method and creating positive and negative training samples. In order to prevent overfitting, it is required to balance the dataset. This drastically increases the number of training sentences. The benefit of this approach is, that it can achieve high performance with low circuit width and circuit depth. The best outcome we could achieve has an accuracy on test data of 82.96%. Keeping circuit width and depth low is helpful for two reasons: In the current era of quantum computing the number of qubits on quantum hardware is severely limited. Also, each additional gate in a circuit introduces additional noise. A binary approach circumvents both of these limitations. The question stands open if the blowup of training data will prevent this approach for application on bigger datasets.

Multiclass classification approach: This strategy deals with masked language modeling in a more conservative way, where sentence labels are more than two-dimensional vectors and represent the masked word. This thesis provides an inside into the development of a masking approach since simply switching all gates of an ansatz to identity gates does not work in some cases. We give a possible approach to masking, which works for all samples, that occur in the dataset. Secondly, we describe a method that

can reduce the number of overall qubits and the task complexity at the same time. This is very much needed, since our evaluation showed, that in order to gain performance in a multiclass classification task, scaling of the model is needed. Right now, due to limitations in classical computing and quantum computing, this scaling is not possible. The best outcome we could achieve managed an accuracy on test data of 34.78%. The masking approach we proposed was successful as well as the strategy for reducing the number of qubits and the task complexity.

We conclude, that in our current era of quantum computing, we are not only limited by computing resources but also in terms of the complexity of our quantum layers. The layers at hand right now, maybe do not have the complexity needed in order to process more complex information. But there are efforts to emulate how information is processed by classical neural networks in quantum circuits [29].
As of the present state, quantum natural language processing has shown good performance in binary classification tasks. However, in the context of multiclass classification more research and advancements are needed in order to complete more complex linguistic tasks. Multiclass classification is essential in real-world NLP tasks. Currently, our research on multiclass classification is constrained by limited computing resources in both classical simulation and real quantum hardware cases for quantum computing. We think that the further deployment of QNLP will rely on the effectiveness of multiclass classification in a quantum environment. Furthermore, the efficiency of QNLP also heavily relies on the development of more sophisticated and powerful quantum hardware. Continued efforts in advancing quantum hardware are essential to unlock the true capabilities of quantum natural language processing. The potential of quantum natural language processing or quantum machine learning to surpass classical methods in the near future remains uncertain.

7.2. Limitations

The practical work of this thesis deals with a quantum machine learning model, that is only trained on 86 sentences and is evaluated on 23 sentences. We do not measure any downstream task performance. The small dataset only provides an inside to the concept of masked language modeling in a quantum machine learning setup and does not provide an inside of how scaling could work in this area. Also, the dataset just contains short sentences. Additionally, only one word per sentence is masked. There

is no inside provided for cases when we want to mask more than one word in a longer sentence. The approaches studied are not tested on complex and long sentences. The occurrence of non-grammatical sentences and out of vocabulary words remains a problem in the DisCoCat framework. The multiclass classification models are only trained on cross entropy loss. More performance could be gained when testing or developing other loss functions for QNLP. For multiclass classification, there is still more work to be done in the exploration of hyperparameters.

7.3. Outlook and further research

Firstly, this work shows how to transform a word prediction task into a binary classification task, that can achieve good results. This is helpful for researchers, who want to explore the possibilities of QNLP in the current era of quantum computing. Also, the thesis provides the reader with a detailed description of how to create a multiclass classification task in the DisCoCat framework. This can be used by researchers in the field to generate other multiclass classification tasks in QNLP. It also shows that a simple identity masking is not possible for all masking occurrences in the DisCoCat framework. The thesis gives a possible solution to that question. We also conceptualize a strategy for reducing the qubit amount and reducing task complexity simultaneously. This strategy extensively utilizes the DisCoCat framework. The strategy has the potential to be applied to various other tasks. We also show that word prediction as a multiclass classification task does not perform very well with the current resources at hand. Overall, these results can be used to extend the multiclass classification research in QNLP. The contributions can also be used as first groundwork regarding masked language modeling with a quantum machine learning setup, maybe even extending it in the future into training quantum language models.
For further research, we would first suggest the development of more ansätze. We think, that the ansätze, that we have right now, are not advanced enough in order to tackle more complex tasks such as word prediction. We would also encourage the research of optimizers for quantum machine learning, in particular, the implementation of an adaptive learning rate into quantum machine learning optimizers. Adaptive learning rate generally will lead to faster convergence, therefore reducing the number of epochs. We also advocate for the implementation of regularization in quantum machine learning. In the future, regularization would make quantum models more resilient against overfitting. In terms of QNLP, we want to suggest exploring word prediction in the newly

developed DisCoCirc [10] framework. We generally want to motivate extensive research of multiclass classification tasks in a quantum machine learning environment using the DisCoCat or DisCoCirc framework. In terms of our experiments, it would be compelling to see how classical machine learning models, without the DisCoCat framework, using roughly the same number of parameters perform at this exact same task in order to make an independent performance comparison. It would be also compelling to see, how the masking approach can be optimized in terms of the number of gates used.

Bibliography

[1] M. Abbaszade, V. Salari, S. S. Mousavi, M. Zomorodi, and X. Zhou. Application of quantum natural language processing for language translation. *IEEE Access*, 9: 130434–130448, 2021. doi: 10.1109/ACCESS.2021.3108768.

[2] J. B. Altepeter, D. F. James, and P. G. Kwiat. *4 Qubit Quantum State Tomography*, pages 113–145. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-44481-7. doi: 10.1007/978-3-540-44481-7_4.

[3] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *J. Mach. Learn. Res.*, 18:153:1–153:43, 2017. URL `http://jmlr.org/papers/v18/17-468.html`.

[4] J. R. Bellegarda. Exploiting latent semantic information in statistical language modeling. *Proc. IEEE*, 88(8):1279–1296, 2000. doi: 10.1109/5.880084.

[5] M. Brandenburg. *Einführung in die Kategorientheorie*. Springer Spektrum, 01 2016. ISBN 978-3-662-47067-1. doi: 10.1007/978-3-662-47068-8.

[6] S. Bruhn. Density matrix methods in quantum natural language processing. Master's thesis, Universität Osnabrück, 2022. doi: 10.48693/111.

[7] D. Camps, R. V. Beeumen, and C. Yang. Quantum fourier transform revisited. *Numer. Linear Algebra Appl.*, 28(1), 2021. doi: 10.1002/nla.2331.

[8] F. Y. Y. Choi, P. M. Wiemer-Hastings, and J. D. Moore. Latent semantic analysis for text segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP 2001, Pittsburgh, PA USA, June 3-4, 2001*. ACL, 2001. URL `https://aclanthology.org/W01-0514/`.

[9] S. Clark. Something old, something new: Grammar-based CCG parsing with transformer models. *CoRR*, abs/2109.10044, 2021. URL `https://arxiv.org/abs/2109.10044`.

[10] B. Coecke. The mathematics of text structure. *CoRR*, abs/1904.03478, 2019. doi: 10.48550/arXiv.1904.03478.

[11] B. Coecke and É. Paquette. *Categories for the Practising Physicist*, pages 173–286. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-12821-9. doi: 10.1007/978-3-642-12821-9_3.

[12] B. Coecke, M. Sadrzadeh, and S. Clark. Mathematical foundations for a compositional distributional model of meaning. *CoRR*, abs/1003.4394, 2010. doi: 10.48550/arXiv.1003.4394.

[13] B. Coecke, G. de Felice, K. Meichanetzidis, and A. Toumi. Foundations for near-term quantum natural language processing. *CoRR*, abs/2012.03755, 2020. doi: 10.48550/arXiv.2012.03755.

[14] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423.

[15] L. Friedrich and J. Maziero. Avoiding barren plateaus with classical deep neural networks. *CoRR*, abs/2205.13418, 2022. doi: 10.48550/arXiv.2205.13418.

[16] J. Gacon, C. Zoufal, G. Carleo, and S. Woerner. Simultaneous perturbation stochastic approximation of the quantum fisher information. *Quantum*, 5:567, 2021. doi: 10.22331/q-2021-10-20-567.

[17] R. B. Griffiths. Hilbert space quantum mechanics is noncontextual. *Studies in History and Philosophy of Science Part B: Studies in History and Philosophy of Modern Physics*, 44(3):174–181, 2013. doi: 10.1016/j.shpsb.2013.02.001.

[18] L. K. Grover. A fast quantum mechanical algorithm for database search. In G. L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219. ACM, 1996. doi: 10.1145/237814.237866.

[19] V. Havlícek, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nat.*, 567(7747):209–212, 2019. doi: 10.1038/s41586-019-0980-2.

[20] C. Heunen and J. Vicary. *Categories for Quantum Theory: An Introduction*. Oxford University Press, 11 2019. doi: 10.1093/oso/9780198739623.001.0001.

[21] C. Heunen, J. Vicary, and D. Reutter. *Categorical Quantum Mechanics: An Introduction*. Oxford University, 2019. URL `https://www.cs.ox.ac.uk/files/10510/notes.pdf`.

[22] Y. Ho and S. Wookey. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE Access*, 8:4806–4813, 2020. doi: 10.1109/ACCESS.2019.2962617.

[23] M. Homeister. *Quantum Computing verstehen*. Computational Intelligence. Springer Fachmedien, Wiesbaden, Germany, 5 edition, 2018. doi: 10.1007/978-3-658-22884-2.

[24] R. A. Horn and C. R. Johnson. *Matrix Analysis, 2nd Ed.* Cambridge University Press, 2012. ISBN 9780521548236. doi: 10.1017/CBO9781139020411.

[25] T. Hubregtsen, J. Pichlmeier, P. Stecher, and K. Bertels. Evaluation of parameterized quantum circuits: on the relation between classification accuracy, expressibility, and entangling capability. *Quantum Mach. Intell.*, 3(1):1–19, 2021. doi: 10.1007/s42484-021-00038-w.

[26] J. Jost. *Kategorientheorie*. Springer Spektrum, Wiesbaden, Germany, 1 edition, 2019. doi: 10.1007/978-3-658-28313-1.

[27] D. Kartsaklis, M. Sadrzadeh, S. Pulman, and B. Coecke. *Reasoning about meaning in natural language with compact closed categories and Frobenius algebras*, page 199–222. Lecture Notes in Logic. Cambridge University Press, 2016. doi: 10.1017/CBO9781139519687.011.

[28] D. Kartsaklis, I. Fan, R. Yeung, A. Pearson, R. Lorenz, A. Toumi, G. de Felice, K. Meichanetzidis, S. Clark, and B. Coecke. lambeq: An Efficient High-Level Python Library for Quantum NLP. *arXiv preprint arXiv:2110.04236*, 2021.

[29] N. Killoran, T. R. Bromley, J. M. Arrazola, M. Schuld, N. Quesada, and S. Lloyd. Continuous-variable quantum neural networks. *CoRR*, abs/1806.06871, 2018. URL `http://arxiv.org/abs/1806.06871`.

[30] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. doi: 10.48550/arXiv.1412.6980.

[31] J. Lambek. *From Word to Sentence: A Computational Algebraic Approach to Grammar*. Open access publications. Polimetrica, 2008. ISBN 9788876991172.

[32] F. W. Lawvere and S. H. Schanuel. *Conceptual mathematics - a first introduction to categories*. Cambridge University Press, 1997. ISBN 978-0-521-47249-4.

[33] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. doi: 10.48550/arXiv.1907.11692.

[34] R. Lorenz, A. Pearson, K. Meichanetzidis, D. Kartsaklis, and B. Coecke. QNLP in practice: Running compositional models of meaning on a quantum computer. *CoRR*, abs/2102.12846, 2021. doi: 10.48550/arXiv.2102.12846.

[35] I. Loshchilov and F. Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017. URL `http://arxiv.org/abs/1711.05101`.

[36] S. MacLane. *Categories for the Working Mathematician*. Springer-Verlag, New York, 1971. Graduate Texts in Mathematics, Vol. 5.

[37] D. Maclaurin, D. Duvenaud, and R. P. Adams. Autograd: Effortless gradients in numpy. In *ICML 2015 AutoML workshop*, volume 238, 2015.

[38] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven. Barren plateaus in quantum neural network training landscapes. *CoRR*, abs/1803.11173, 2018.

[39] K. Meichanetzidis, S. Gogioso, G. de Felice, N. Chiappori, A. Toumi, and B. Coecke. Quantum natural language processing on near-term quantum computers. *Electronic Proceedings in Theoretical Computer Science*, 340:213–229, 2021. doi: 10.4204/eptcs.340.11.

[40] K. Meichanetzidis, A. Toumi, G. de Felice, and B. Coecke. Grammar-aware sentence classification on quantum computers. *Quantum Machine Intelligence*, 5(1), 2023. doi: 10.1007/s42484-023-00097-1.

[41] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013. doi: 10.48550/arXiv.1310.4546.

[42] I. C. Mogotsi, C. D. Manning, P. Raghavan, and H. Schütze. Introduction to information retrieval. *Inf. Retr.*, 13(2):192–195, 2010. doi: 10.1007/s10791-009-9115-y.

[43] A. Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 78, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138385. doi: 10.1145/1015330.1015435.

[44] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge University Press, 2016. ISBN 978-1-10-700217-3.

[45] Qiskit contributors. Qiskit: An open-source framework for quantum computing, 2023. URL `https://learn.qiskit.org/course/machine-learning/introduction`.

[46] V. V. Raghavan and S. K. M. Wong. A critical analysis of vector space model for information retrieval. *J. Am. Soc. Inf. Sci.*, 37(5):279–287, 1986. doi: 10.1002/(SICI)1097-4571(198609)37:5⟨279::AID-ASI1⟩3.0.CO;2-Q.

[47] B. Rodatz. Conversational negation in compositional distributional semantics. Master's thesis, University of Oxford, 2021. URL `https://www.cs.ox.ac.uk/people/bob.coecke/BenThesis`.

[48] D. Rothman and A. Gulli. *Transformers for Natural Language Processing: Build, train, and fine-tune deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, and GPT-3*. Packt Publishing, 2022. ISBN 9781803243481.

[49] W. Rudin. *Real and Complex Analysis*. Mathematics series. McGraw-Hill, 1987. ISBN 9780071002769.

[50] P. Sadegh and J. C. Spall. Optimal random perturbations for stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Trans. Autom. Control.*, 43(10):1480–1484, 1998. doi: 10.1109/9.720513.

[51] M. Sadrzadeh, D. Kartsaklis, and E. Balkir. Sentence entailment in compositional distributional semantics. *Ann. Math. Artif. Intell.*, 82(4):189–218, 2018. doi: 10.1007/s10472-017-9570-x.

[52] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3), mar 2019. doi: 10.1103/physreva.99.032331.

[53] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, 101(3), mar 2020. doi: 10.1103/physreva.101.032308.

[54] H. Schütze. Automatic word sense discrimination. *Comput. Linguistics*, 24(1): 97–123, 1998. ISSN 0891-2017. URL `https://aclanthology.org/J98-1004`.

[55] R. A. Shaikh, L. Yeh, B. Rodatz, and B. Coecke. Composing conversational negation. *Electronic Proceedings in Theoretical Computer Science*, 372:352–367, 2022. doi: 10.4204/eptcs.372.25.

[56] S. Sim, P. D. Johnson, and A. Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, oct 2019. doi: 10.1002/qute.201900070.

[57] R. S. Sinha and R. Mihalcea. Explorations in lexical sample and all-words lexical substitution. *Nat. Lang. Eng.*, 20(1):99–129, 2014. doi: 10.1017/S1351324912000265.

[58] A. Skolik, J. R. McClean, M. Mohseni, P. van der Smagt, and M. Leib. Layerwise learning for quantum neural networks. *CoRR*, abs/2006.14904, 2020.

[59] A. Toumi and A. Koziell-Pipe. Functorial language models. *CoRR*, abs/2103.14411, 2021. doi: 10.48550/arXiv.2103.14411.

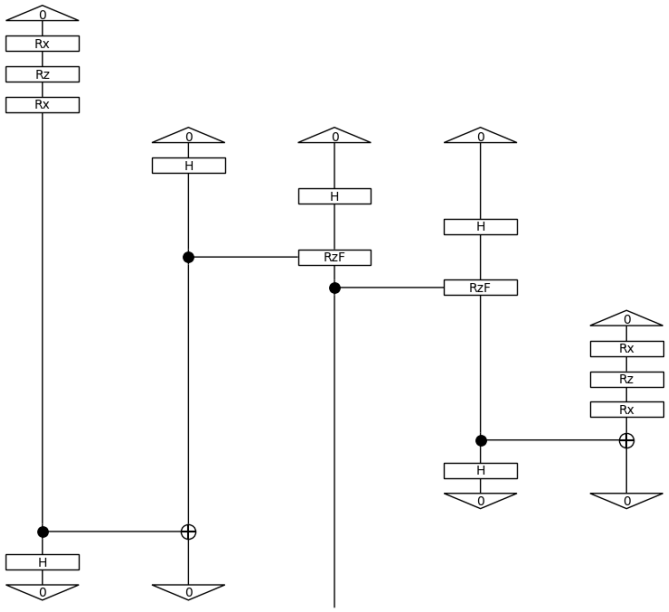
[60] S. Vajjala, B. Majumder, A. Gupta, and H. Surana. *Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems*. O'Reilly Media, 2020. ISBN 9781492054023.

[61] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html`.

Appendices

A. Theoretical insights for the fixed parameter masking

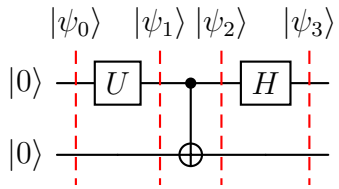
In this section, we want to give a theoretical insight into how the fixed parameter masking works. For this explanation, we will focus again on the test sentence *Alice loves Bob*, where we mask *loves*, since it carries the output wire.



Firstly, we need to clarify how information travels from one qubit to another. This is necessary since the word *Alice* for example, impacts the whole meaning of the sentence. *Alice* is only represented on the first qubit, but we measure the third qubit, which carries the whole representation of the sentence. Secondly, we need to clarify how postselection impacts our measurement probabilities in general. This explanation is needed since we only measure one qubit of the *loves* box to get the entire representation of the sentence. We measure in particular the second out of three qubits of the *loves* box, but the meaning

from *Alice* only travels to the first one. It is also needed for the impact of *Bob* on the entire sentence.

We slowly start with the first one. In terms of category theory, the travel of information from one box to another might be already clear. The cup transforms the output wire of one box to the input wire for another box. But the situation is intriguing from a quantum computing perspective. We will consider the most simple circuit, that represents the mechanisms of a cup:



The U gate represents an arbitrary rotation, which is induced by the R_x R_z R_x procedure in our example.

$$|\psi_0\rangle = |00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$|\psi_1\rangle = (U \otimes I) * |\psi_0\rangle = (\begin{bmatrix} e^{i(\alpha-\frac{\beta}{2}-\frac{\delta}{2})}cos\frac{\gamma}{2} & -e^{i(\alpha-\frac{\beta}{2}+\frac{\delta}{2})}sin\frac{\gamma}{2} \\ e^{i(\alpha+\frac{\beta}{2}-\frac{\delta}{2})}sin\frac{\gamma}{2} & e^{i(\alpha+\frac{\beta}{2}+\frac{\delta}{2})}cos\frac{\gamma}{2} \end{bmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}) * \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$|\psi_1\rangle = \begin{pmatrix} e^{i(\alpha-\frac{\beta}{2}-\frac{\delta}{2})}cos\frac{\gamma}{2} \\ 0 \\ e^{i(\alpha+\frac{\beta}{2}-\frac{\delta}{2})}sin\frac{\gamma}{2} \\ 0 \end{pmatrix}$$

$$|\psi_2\rangle = CNOT * |\psi_1\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} * \begin{pmatrix} e^{i(\alpha-\frac{\beta}{2}-\frac{\delta}{2})}cos\frac{\gamma}{2} \\ 0 \\ e^{i(\alpha+\frac{\beta}{2}-\frac{\delta}{2})}sin\frac{\gamma}{2} \\ 0 \end{pmatrix} = \begin{pmatrix} e^{i(\alpha-\frac{\beta}{2}-\frac{\delta}{2})}cos\frac{\gamma}{2} \\ 0 \\ 0 \\ e^{i(\alpha+\frac{\beta}{2}-\frac{\delta}{2})}sin\frac{\gamma}{2} \end{pmatrix}$$

$$|\psi_3\rangle = (H \otimes I) * |\psi_2\rangle = (\frac{1}{\sqrt{2}} * \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}) * \begin{pmatrix} e^{i(\alpha-\frac{\beta}{2}-\frac{\delta}{2})}cos\frac{\gamma}{2} \\ 0 \\ 0 \\ e^{i(\alpha+\frac{\beta}{2}-\frac{\delta}{2})}sin\frac{\gamma}{2} \end{pmatrix}$$

$$|\psi_3\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} e^{i(\alpha-\frac{\beta}{2}-\frac{\delta}{2})}cos\frac{\gamma}{2} \\ e^{i(\alpha+\frac{\beta}{2}-\frac{\delta}{2})}sin\frac{\gamma}{2} \\ e^{i(\alpha-\frac{\beta}{2}-\frac{\delta}{2})}cos\frac{\gamma}{2} \\ -e^{i(\alpha+\frac{\beta}{2}-\frac{\delta}{2})}sin\frac{\gamma}{2} \end{pmatrix}$$

To calculate the measurement probabilities of the first and the second qubit, we need to first calculate the reduced density matrices. With the reduced density matrices we calculate the measurement probabilities with the help of the measurement operators.

$$\rho = |\psi_3\rangle \langle\psi_3|$$

$$\rho = \begin{pmatrix} \frac{cos^2\frac{\gamma}{2}}{2} & \frac{e^{i(\alpha-\frac{\beta}{2}-\frac{\delta}{2})}e^{-i(\alpha+\frac{\beta}{2}-\frac{\delta}{2})}sin\frac{\gamma}{2}cos\frac{\gamma}{2}}{2} & \frac{cos^2\frac{\gamma}{2}}{2} & -\frac{e^{i(\alpha-\frac{\beta}{2}-\frac{\delta}{2})}e^{-i(\alpha+\frac{\beta}{2}-\frac{\delta}{2})}sin\frac{\gamma}{2}cos\frac{\gamma}{2}}{2} \\ \frac{e^{-i(\alpha-\frac{\beta}{2}-\frac{\delta}{2})}e^{i(\alpha+\frac{\beta}{2}-\frac{\delta}{2})}sin\frac{\gamma}{2}cos\frac{\gamma}{2}}{2} & \frac{sin^2\frac{\gamma}{2}}{2} & \frac{e^{-i(\alpha-\frac{\beta}{2}-\frac{\delta}{2})}e^{i(\alpha+\frac{\beta}{2}-\frac{\delta}{2})}sin\frac{\gamma}{2}cos\frac{\gamma}{2}}{2} & -\frac{sin^2\frac{\gamma}{2}}{2} \\ \frac{cos^2\frac{\gamma}{2}}{2} & \frac{e^{i(\alpha-\frac{\beta}{2}-\frac{\delta}{2})}e^{-i(\alpha+\frac{\beta}{2}-\frac{\delta}{2})}sin\frac{\gamma}{2}cos\frac{\gamma}{2}}{2} & \frac{cos^2\frac{\gamma}{2}}{2} & -\frac{e^{i(\alpha-\frac{\beta}{2}-\frac{\delta}{2})}e^{-i(\alpha+\frac{\beta}{2}-\frac{\delta}{2})}sin\frac{\gamma}{2}cos\frac{\gamma}{2}}{2} \\ -\frac{e^{-i(\alpha-\frac{\beta}{2}-\frac{\delta}{2})}e^{i(\alpha+\frac{\beta}{2}-\frac{\delta}{2})}sin\frac{\gamma}{2}cos\frac{\gamma}{2}}{2} & -\frac{sin^2\frac{\gamma}{2}}{2} & -\frac{e^{-i(\alpha-\frac{\beta}{2}-\frac{\delta}{2})}e^{i(\alpha+\frac{\beta}{2}-\frac{\delta}{2})}sin\frac{\gamma}{2}cos\frac{\gamma}{2}}{2} & \frac{sin^2\frac{\gamma}{2}}{2} \end{pmatrix}$$

$$\rho_1 = Tr_2(\rho) = \begin{pmatrix} \frac{sin^2\frac{\gamma}{2}+cos^2\frac{\gamma}{2}}{2} & \frac{-sin^2\frac{\gamma}{2}+cos^2\frac{\gamma}{2}}{2} \\ \frac{-sin^2\frac{\gamma}{2}+cos^2\frac{\gamma}{2}}{2} & \frac{sin^2\frac{\gamma}{2}+cos^2\frac{\gamma}{2}}{2} \end{pmatrix}$$

$$\rho_2 = Tr_1(\rho) = \begin{pmatrix} cos^2\frac{\gamma}{2} & 0 \\ 0 & sin^2\frac{\gamma}{2} \end{pmatrix}$$

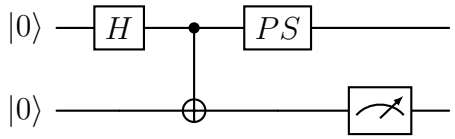
The measurement probabilities of both qubits therefore are:

$$p(0)_{\rho_1} = Tr(M_0^\dagger M_0 \rho_1) = \frac{sin^2\frac{\gamma}{2} + cos^2\frac{\gamma}{2}}{2}$$
$$p(1)_{\rho_1} = Tr(M_1^\dagger M_1 \rho_1) = \frac{sin^2\frac{\gamma}{2} + cos^2\frac{\gamma}{2}}{2}$$

$$p(0)_{\rho_2} = Tr(M_0^\dagger M_0 \rho_2) = cos^2\frac{\gamma}{2}$$
$$p(1)_{\rho_2} = Tr(M_1^\dagger M_1 \rho_2) = sin^2\frac{\gamma}{2}$$

We can see, that the probability of the first qubit for measuring 0 or 1 is always exactly 0.5, no matter what the rotation angle beforehand might be. Every γ angle in $\frac{sin^2\frac{\gamma}{2}+cos^2\frac{\gamma}{2}}{2}$ always return 0.5.
The second qubit has the exact probability amplitudes, which the first qubit has, after the application of the U gate. It is shown that the cup transfers the information from one qubit to the other. In other words, it transfers the meaning of one box to another box.

Now we move to the second part, where we want to present how the postselection criterion on one qubit, can influence the measurement probabilities of another qubit. We want to showcase that phenomenon on the following simple circuit, where the PS gate symbolizes the postselection on any measurement condition:

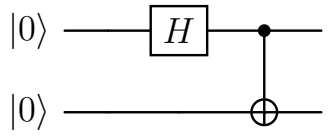
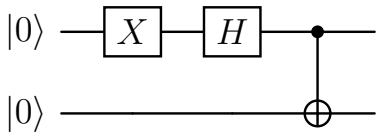
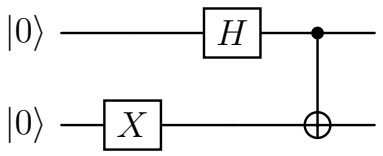
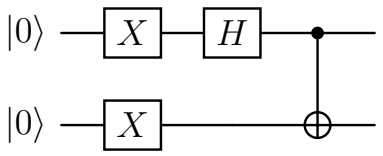


When we have no postselection criterion on qubit 1, qubit 2 possesses: $|\alpha|^2 = 0.5$ and $|\beta|^2 = 0.5$.
When we introduce a postselection on qubit 1 with a 0-effect, qubit 2 possesses: $|\alpha|^2 = 1$ and $|\beta|^2 = 0$.
On the other hand, when we introduce a postselection on qubit 1 with a 1-effect, qubit 2 possesses: $|\alpha|^2 = 0$ and $|\beta|^2 = 1$.

With this simple circuit, we see that postselection on one qubit can directly impact the measurement probabilities of the second qubit. But what does this mean for the fixed parameter masking approach?
In general, the goal of masked language modeling is to adjust the parameters of the model in a way so that they can predict the masked word. Combining these two phenomenons, which we explained, it is possible to see that with the fixed parameters approach it is possible to do that in a quantum machine learning setup, especially for the case, where the masked word and the output wires overlap. Firstly, always remember, that the masked word does have random parameters, but they do not change over the course of the training. With the first phenomenon, we see that information can travel from one qubit to the other without getting lost. That means we influence the probability distribution of the qubits with the fixed parameters. Secondly, through the postselection criterion, the influence of the traveled information from one word to another has an impact on the output wires, even though they might not be connected. This is particularly important for the case, where output wires and masking overlap. By essentially connecting all qubits with controlled R_z gates, the random state of the masked word can be influenced by the non-masked words in order to generate a valid prediction.

B. Bell states

Here, the reader can find all of the four Bell states with a corresponding circuit:

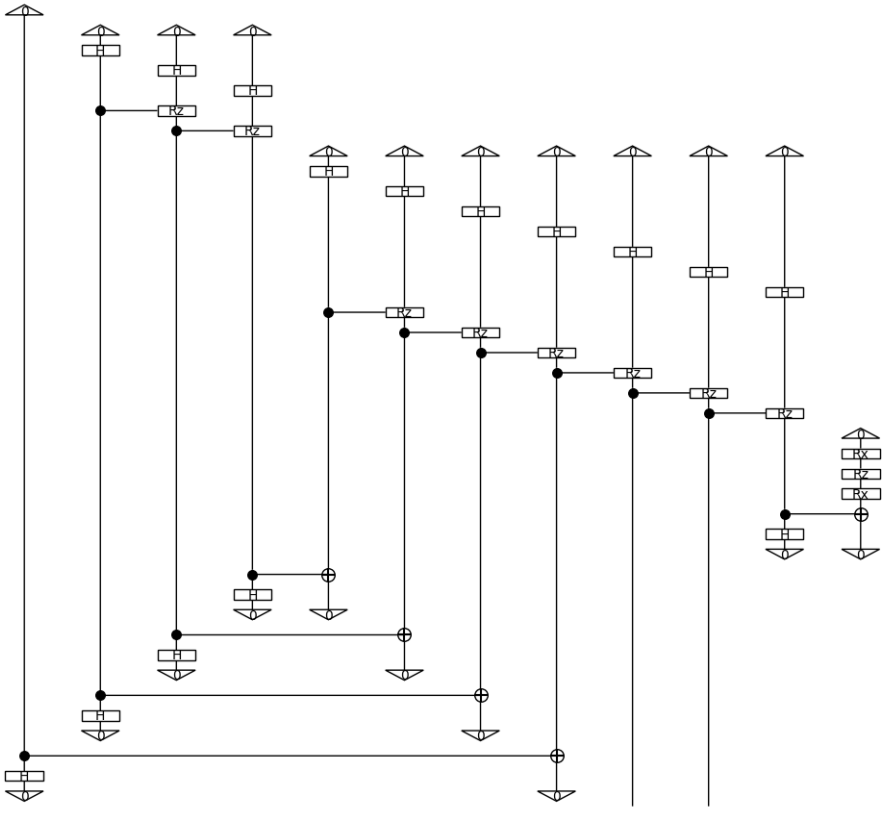
Label	Quantum State	Quantum circuit
Φ^+	$\frac{1}{\sqrt{2}}(\|00\rangle + \|11\rangle)$	
Φ^-	$\frac{1}{\sqrt{2}}(\|00\rangle - \|11\rangle)$	
Ψ^+	$\frac{1}{\sqrt{2}}(\|01\rangle + \|10\rangle)$	
Ψ^-	$\frac{1}{\sqrt{2}}(\|01\rangle - \|10\rangle)$	

C. Circuits

In this section, complete circuits of the examples shown in the main body are displayed.

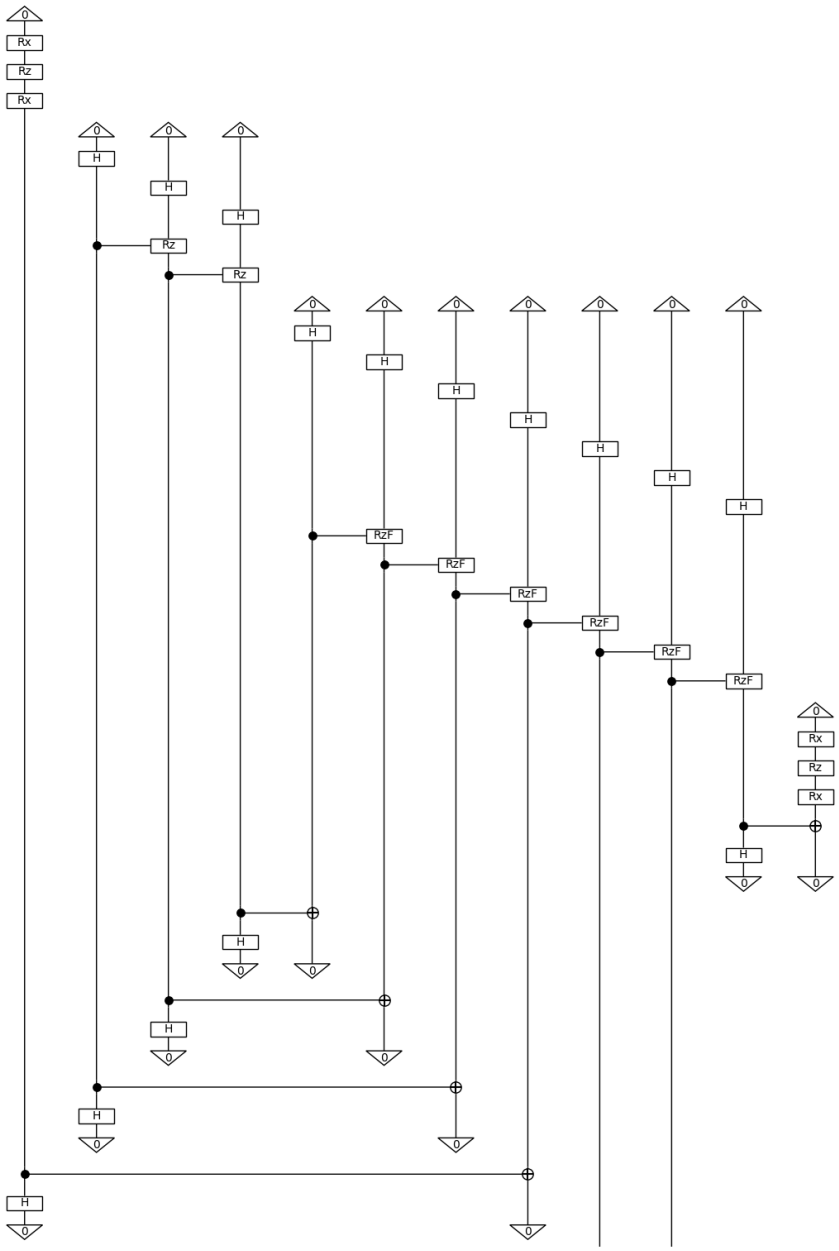
C.1. Circuit for identity masking

The circuit of the sentence *cat runs on land* for a multiclass classification task, where the first word *cat* is masked with identity masking:



C.2. Circuit for fixed parameter masking

The circuit of the sentence *cat runs on land* for a multiclass classification task, where the first word *on* is masked with fixed parameter masking:



D. Dataset

The Dataset used in both experiments was taken from [59]. The integer at the end of every sentence indicates the position of the masked word in the sentence:

fox runs after chicken 0
mouse runs on land 3
seal eats fish 2
cat eats fish 2
dog runs after fox 0
dog runs after fox 3
dog runs on land 3
seal swims 0
mouse squeaks 0
chicken eats grain 0
fox chases chicken 2
seal runs on land 3
cat chases mouse 2
cat runs on land 3
dog runs after cat 0
cat runs after mouse 3
whale swims in water 3
mouse flees cat 2
dog eats bone 2
fox runs on land 3
fox chases after chicken 0
seal swims in water 0
fish swims in water 3
cat chases after mouse 3
fox runs after chicken 3
fox bites chicken 2
dog barks at cat 0
fox chases 0
mouse bites cheese 2
dog chases after cat 0
dog barks 0
dog chases after fox 0
cat chases after mouse 0
chicken clucks 0
mouse eats cheese 2
cat bites fish 2
chicken flees fox 2
fox eats chicken 2
cat meows 0
dog chases fox 2
seal runs 0
cat flees dog 2
seal eats fish 1

mouse eats cheese 1
dog bites fox 1
whale eats krill 1
dog eats bone 1
fox eats chicken 1
chicken eats grain 1
cat flees dog 1
mouse flees cat 1
cat bites mouse 1
fox flees dog 1
chicken clucks 1
dog barks 1
chicken runs on land 1
whale swims 1
mouse runs on land 1
dog barks at cat 1
seal runs on land 1
cat runs on land 1
whale swims in water 1
dog barks at fox 1
dog runs on land 1
dog chases after fox 1
fox chases 1
seal runs 1
seal swims in water 1
cat chases after mouse 1
fox runs on land 1
dog chases after cat 1
seal runs on land 2
seal swims in water 2
fish swims in water 2
fox runs on land 2
fox chases after chicken 2
fox runs after chicken 2
whale swims in water 2
dog runs after cat 2
cat runs after mouse 2
dog runs after fox 2
dog barks at fox 2
dog barks at cat 2
dog runs on land 2
chicken runs on land 2
dog chases after cat 2

cat runs after mouse 0
seal runs on land 0
dog chases after fox 3
dog barks at fox 0
whale eats krill 2
fox flees dog 2
dog barks at fox 3
seal swims in water 3
chicken runs on land 3
dog bites bone 2
fox chases after chicken 3
cat eats fish 1
dog bites cat 1
chicken flees fox 1
mouse squeaks 1
cat meows 1
fish swims 1
fox chases after chicken 1
fish swims in water 1
cat chases after mouse 2
mouse runs on land 2
cat runs on land 2
dog chases after fox 2

E. Technical limitations

Training quantum machine learning models on classical machines requires a substantial amount of computing memory due to the simulation of quantum circuits involved. Therefore, scaling the quantum models demands a significant amount of memory resources. In the multiclass classification case, we wanted to compute one model with configuration: $q_n = 3$; $q_p = 3$; $n_{layers} = 13$. This was at first not possible, since in the first epochs the package numpy needed to allocate 96 GB of RAM for one array. Consequently, we scaled our classical system in order to accommodate those large data structures and let the model train again. Unfortunately after a few epochs again, a similar error occurred, where numpy would need to allocate 1 TB or more memory to one array. Our classical hardware reached its limitations and could no longer cope with the demands posed by handling such large data types.

F. Loss functions

Over the course of training two different loss functions are used. This appendix section will shortly describe them. In the experiment, there is a small distinction between binary cross entropy and cross entropy. This lies in the fact that cross entropy can be shortened when having only two classes.

Cross entropy:

$$L = -\frac{1}{N}\sum_{i=1}^{N} t_i \cdot \log(p_i)$$

Where t_i is the true label and p_i is the predicted probability for one class. N is the number of classes.

Binary Cross entropy:

$$\begin{aligned} L &= -\frac{1}{2}\sum_{i=1}^{2} t_i \cdot \log(p_i) \\ &= -\frac{1}{2}(t_1 \cdot \log(p_1) + t_2 \cdot \log(p_2)) \\ &= -\frac{1}{2}(t \log(p) + (1-t) log(1-p)) \end{aligned}$$

Where t is the true label and p is the predicted probability for one class as well.

Hinge loss:

$$L = \sum_{i=1}^{2} max(0, 1 - t_i \cdot p_i)$$

$t_i \pm 1$ is the intended output and p_i refers to our predicted value. This function is intended for binary classification.

G. Verification of the yanking equations

In this appendix section, we want to verify the yanking equation with the cup being represented by the Bell measurement and the cap being represented by the Bell state. We start with the first one from section 4.6:

$$
\begin{aligned}
(\epsilon_A \otimes id_A) \circ (id_A \otimes \eta_A) &= (\frac{1}{\sqrt{2}}(\langle 00| + \langle 11|) \otimes I) * (I \otimes (\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)) \\
&= \frac{1}{\sqrt{2}}(\begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}) * \frac{1}{\sqrt{2}}(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}) \\
&= \frac{1}{2}(\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}) \\
&= \frac{1}{2} * \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I = id_A
\end{aligned}
$$

Now the second one:

$$
\begin{aligned}
(id_A \otimes \epsilon_A) \circ (\eta_A \otimes id_A) &= \left(I \otimes \frac{1}{\sqrt{2}}(\langle 00| + \langle 11|) \right) * \left(\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \otimes I \right) \\
&= \frac{1}{2} \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix} \right) * \left(\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\
&= \frac{1}{2} \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \\
&= \frac{1}{2} * \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I = id_A
\end{aligned}
$$

We can conclude that the yanking equation can be verified using this representation of caps and cups. It is important to note that the effects depicted in the cups are not just measurements, but measurements with postselection on the 0-effect. Consider section 4.3.2 for further details on the concept of effects and postselection.

H. Abbreviations

BERT	Bidirectional Encoder Representations from Transformers
DisCoCat	Categorical Compositional Distributional (framework)
DisCoCirc	Circuit-shaped Compositional Distributional (framework)
FHilb	Category Of Finite Dimensional Hilbert spaces
IQP	Instantaneous Quantum Polynomial (ansatz)
MC	Meaning Classification (task)
MLM	Masked Language Modeling
NISQ	Noisy Intermediate-Scale Quantum (hardware)
NLP	Natural Language Processing
QC	Quantum Computing
QML	Quantum Machine Learning
QNLP	Quantum Natural Language Processing
RoBERTa	A Robustly Optimized BERT Pretraining Approach
RP	Relative Phrase (classification task)
SIM 14	14th ansatz from Sukin Sim et al. [56]
SIM 15	15th ansatz from Sukin Sim et al. [56]
SPSA	Simultaneous Perturbation Stochastic Approximation

Declaration

I hereby declare that this master's thesis is my own work, I have marked all citations and I have documented all sources and materials used.

Munich, 31.07.2023

.....
Jakob Murauer