NExt ApplicationS of Quantum Computing

D6.1 QNLP design and specification

Document Properties

Contract Number	951821
Contractual Deadline	28/02/2021
Dissemination Level	Public
Nature	Report
Edited by:	Antonio Villalpando, ICHEC Lee J. O'Riordan, ICHEC Kaspars Balodis, Tilde Rihards Krišlauks, Tilde
Authors	Antonio Villalpando, ICHEC Lee J. O'Riordan, ICHEC Kaspars Balodis, Tilde Rihards Krišlauks, Tilde
Reviewers	Vedran Dunjko, ULEI Andrés Gómez, CESGA
Date	24/02/2021
Keywords	Natural Language Processing, Quantum Computing, Tensor Networks
Status	Delivered
Release	1.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 951821

History of Changes

Release	Date	Author, Organization	Description of Changes
0.2	01/02/2021	Antonio Villalpando, ICHEC Lee J. O'Riordan, ICHEC Kaspars Balodis, Tilde Rihards Krišlauks, Tilde	Sections 2 and 3, including background, motivation, as well as software architecture and package testing proposals.
1.0	24/02/2021	Antonio Villalpando, ICHEC Lee J. O'Riordan, ICHEC Kaspars Balodis, Tilde Rihards Krišlauks, Tilde	Final revisions of reviewer comments

Table of Contents

1. Executive Summary

Understanding the applicability of NISQ-era devices for a variety of problems is of the utmost importance to better develop and utilise these devices for real-world use-cases. In this document we motivate the use of quantum computing models for natural-language processing tasks, focussing on comparison with existing methods in the classical natural language processing (NLP) community. We define the current state of these NISQ devices, and define methods of interest that will allow us to exploit the resources to implement NLP tasks, by encoding and processing data in a hybrid classical-quantum workflow. For this, we outline the high-level architecture of the solution, and provide a modular design for ease of implementation and extension.

2. Context

2.1. QNLP project introduction

The goal of the QNLP project is to define the required processes and mappings for natural language processing tasks on quantum computing devices. Given the infancy of quantum technologies, allowing for the exploration of valid methods and new algorithms to exploit the computational space these platforms offer is essential. Natural language processing (NLP) in the language of quantum mechanics has seen a large amount of work in recent years (Coecke, Felice, Meichanetzidis, & Toumi, 2020) (Coecke, Sadrzadeh, & Clark, 2010) (O'Riordan, Doyle, Baruffa, & Kannan, 2020), and in this deliverable we will define the steps to explore the solution of NLP tasks on near-term quantum devices.

Here, we aim to define the background material, and current state of quantum hardware and architectures in the noisy-intermediate scale quantum (NISQ) device era (Preskill, 2018).This motivates the operations and processes that can be defined to work on such noisy machines, of which will differ from fully fault-tolerant devices. Following this, we will discuss the current state of (classical) NLP methods, from the point of view of techniques, uses and challenges.

Given these two preceding sections, we infer the best algorithms that can be built using the NISQ-compatible methods for NLP, motivated by the current state-of-the-art methods and theories available.

2.2. Quantum background

2.2.1. Current state of quantum computing

Advances in qubit architecture, noise control, and the proliferation of quantum algorithms has led to the exploration of a variety of new topics for quantum devices. With several platforms in the cloud, and more on-premise in research labs worldwide, end-users can interact with quantum computers via a variety of interfaces, as well as carrying out simulations through highly used programming languages such as Python, C++ and Julia. With this growth in accessibility, researchers from all over the world can access the latest quantum devices and use them in their work. As a result, quantum computing has found immediate application or a faster assimilation in fields outside their original use-cases.

Current generation gate based quantum computers have on order of 10s of qubits, which are expected to dramatically increase in the coming years (IBM, 2020) (IonQ, 2020). This, in addition to the fact that the circuit depths and coherence times are still restrictive, means we are yet far to achieve fault-tolerant quantum computing, which would allow to experimentally confirm the speedup of certain algorithms such as Shor's algorithm, run advanced quantum chemistry computations, or carry out other practical applications. Instead, we are now in the so-called Noisy Intermediate-Scale Quantum (NISQ) era (Preskill, 2018), and while the aforementioned applications are still not available, a different approach to quantum computing has been proposed.

The idea for the current devices is to use them in tandem with classical machines in hybrid algorithms. Classical optimization is made on the classical computers, as current quantum devices cannot outperform classical machines in this task. On the other hand, state preparation is left to quantum computer to leverage the unique properties of quantum states. This approach is currently undergoing a significant research investment, and is being widely explored for applications in the field known as quantum machine learning (QML), as well as in academic research, and we can mention as an example the successful application of quantum algorithms in computational chemistry (McArdle, 2020).

2.2.2. Available quantum devices

The circuit model, consisting of sequential application of reversible quantum transformations acting on a qubit register, is the dominant model in quantum computing. Several qubit technologies have been developed in the last decades, with different performance regarding noise, coherence time, etc. Among these technologies, the most common are superconducting (Kjaergaard, et al., 2020) (charge, flux or phase) qubits made out of Josephson junctions, photon polarization qubits, spin qubits or quantum dots. Ion or atom arrays is another qubit architecture that has been recently implemented with great expectations for the forthcoming years. Usually magnetic fields are use to manipulate them and gates are implemented through the use of microwaves or lasers.

Among the pioneering companies building quantum computers, stand out IBM, with a 65 superconducting qubits quantum computer and an announced 1000-qubit machine by 2023 (Gambetta, 2020).

Other companies such as Rigetti or Xanadu offer cloud computing services operating quantum computers. This kind of services is having a rapid growth, and we can mention for example the recently announced Azure Quantum, a collaboration of Microsoft with partners such as IonQ or Honeywell, both companies working with trapped-ion quantum computers.

Some European companies are also putting effort into the development of quantum computing. AQT, for example, use ion-trap quantum computers with single charged atoms as qubits trapped inside vacuum chambers. They were able to implement a scalable Shor algorithm (Monz, 2016), or perform topologically encoded quantum error correction (Nigg, 2014), among other achievements. Furthermore, they are actively developing a 50-qubit quantum computing demonstrator based on trapped ions, a technology where temperature is not as restrictive as for superconducting qubits (Pogorelov, 2021). Pasqal, on the other hand, explores quantum simulators and works towards the commercialization of neutral atoms-based hardware, that can reach quantum registers with a larger number of qubits and higher connectivity (Henriet, 2020).

One point of note is that all of these devices are currently well within the realm of NISQ, wherein errors inherent in the devices prevent their use for large-depth quantum circuits.

In addition to these, alternative models of computation have been proposed and well studied, such as quantum annealers, boson samplers, both of which are not universal quantum computers, and topological quantum computers, which if experimentally realised have built in fault tolerance. In fact, a quantum annealer has been the first quantum device to be commercially distributed by D-Wave. However, a general purpose error-corrected quantum computer is still far from realised, and so we restrict ourselves to exploration of NISQ era devices, such as those previously mentioned.

2.2.3. Tensor Networks

Tensor networks (TN) have been widely used in the field of solid state physics for simulating many-body systems, and connections have been made between the structure of such networks and certain classes of quantum states (White, 1993) (Östlund, (1995)). The Penrose graphical notation helped to depict these networks visually (Penrose, 1971), where akin to Feynman diagrams, one can form computational graphs with this visual notation. In many-body quantum physics, high-dimensional tensors appear often, with the number of indices scaling exponentially with the number of constituents of the system. Tensor networks provide efficient representation of these kinds of states, rewriting them as many low-dimensional tensors that carry the quantum properties such as the entanglement in the bond dimension of the contracted indices, allowing results to be computed in a more efficient way. This view not only gives new insight into the understanding of complex quantum systems but also helps to develop new simulation algorithms.

A key idea working with TN is the singular value decomposition (SVD), which allows to approximate a wavefunction keeping the relevant degrees of freedom responsible for the entanglement of the state. A popular method is the Density Matrix Normalization Group (DMRG) (Schollwöck, 2011), an

optimization algorithm that produces 1-d finitely-correlated states known as Matrix Product States (MPS) (Orús, 2014). This compact way to represent complex systems and the underlying compression of high-dimensional data also finds application in deep learning and neural network representation (A. Cichocki, 2016) (Andrey Kardashin, 2018).

On the other hand, most of the time, quantum algorithms are depicted as quantum circuits, which are diagrams inspired by tensor network theory (Biamonte & Bergholm, 2017). In fact, it is possible to implement tensor network models in small NISQ devices (Huggins, Patil, Mitchell, Whaley, & Stoudenmire, 2019) (Wall, Abernathy, & Quiroz, 2010) as the number of physical qubits required can be smaller than the desired output size (Bai, Yang, & Chiribella, 2020), having a logarithmic or even constant scaling for the right architecture choice of the network. With this approach, quantum states are represented by MPS, and linear maps by Matrix Product Operators (MPO) (Chan, Keselman, Nakatani, Li, & White, 2016).

A special use of tensor networks is the ZX-calculus (Wetering, 2020). From a TN point of view, ZX-calculus considers every element of a quantum circuit as a tensor and the wires as the contracted indices. It is especially useful for reasoning using Penrose diagrams (Penrose, 1971). For quantum NLP, ZX-Calculus is interesting for two reasons: parametrized circuits arising in some models are easier to understand in diagrammatic notation, giving us an intuition of the processes involved and also it is extremely useful because it can reduce the depth of quantum circuits (Kissinger & van de Wetering, 2020) which is usually an issue given the limitations of NISQ devices.

2.2.4. Computational complexity and quantum computers

Given the generality of classical computers, the use of quantum computers must allow for some benefits to be considered a worthwhile endeavour. As such, the main area of benefit for these devices is in improved algorithmic complexity for specific tasks. Grover-like methods show a quadratic improvement in database search, Shor's algorithm demonstrates polynomial rather than sub-exponential scaling for integer factorisation, and heuristic variational algorithms accurately match energy levels of molecular systems which require exponential scaling to solve classically.

These complexity improvements show only a small subset of advantages provided by using quantum devices. However, it is worth noting that often the overhead in classically preparing information for these devices can often undo any gains obtained, and so care must be taken when designing algorithms for use on quantum devices. In addition, many algorithms require circuit depths making the use of NISQ devices infeasible due to the strength of noise relative to the signal (Preskill, 2018).

Of the above methods, the variational class of algorithms is known to be more resilient to noise and device imperfections, given the short depths often observed with such circuits (Swingle, 2017). We will consider these methods as part of our proposed design for this reason.

2.2.5. Quantum machine learning & variational circuits

Variational quantum algorithms are, at their core, methods which are defined by parametrized quantum circuits: tunable unitary gates controlled by free parameters acting on the different qubits of our register, and consisting of sequential rotations along the required axis of the Bloch sphere (McClean, Romero, Babbush, & Aspuru-Guzik, 2016). Two popular examples are the variational quantum eigen-solver (VQE) (Peruzzo, et al., 2014) (Cerezo, Sharma, Arrasmith, & Coles, 2020) and the quantum approximate optimization algorithm (QAOA) (Farhi, Goldstone, & Gutmann, 2014). The states for these methods are prepared in a convenient Ansätz that satisfy some requirements of the problem to be solved and hardware architecture (Vicentini, Biella, Regnault, & Ciuti, 2019) (Choquette, et al., 2020) (Herasymenko & O'Brien, 2019). These Ansätze are specially developed for modelling complex interactions, and find favour in quantum chemistry and molecular simulations (Grimsley,

Economou, Barnes, & Mayhall, 2019). The goal is to compute the value of an objective function using the quantum computer and redefine the parameters for the next run using a classical optimizer.

Another approach to optimize PQCs are differentiable quantum circuits (Bergholm, et al., 2018), where the objective function is evaluated for a range of different values around the input parameter, which in turn determines the new set of parameters for the next iteration. This “quantum differentiation” has been proposed to solve classical deep learning problems such as the vanishing gradient. This multilayer picture can be compared to that of neural networks, where the free parameters of the circuits are equivalent to the weight and biases of the network (Bharti, 2021).

2.3. Classical NLP

The field of natural language processing (NLP) deals with processing human language, including speech recognition, natural language understanding (NLU), and natural language generation. NLU deals with comprehension and understanding of human language and extracting structured knowledge from textual or speech input. Historically, many of the NLP methods have employed rule-based approaches (Danilevsky, u.c., 2020). Although such methods provide direct means to apply the domain knowledge of experts, and the outputs of the respective models are, in principle, interpretable, they often prove to be difficult to scale up due to the open-ended nature of language.

Nowadays, although rule-based methods are still used in NLP, many subfields of NLP have largely shifted to using Machine Learning (ML) based methods (Jurafsky & Martin, 2017). Although ML methods oftentimes suffer in regard to model interpretability (Danilevsky, u.c., 2020), they propose a more scalable approach to solving NLP tasks. ML methods are used to a great extent in recent research efforts in the NLP field. In real-world applications of NLP methods, such as intent detection and parallel data extraction, typically, ML methods, such as word embeddings and language models, are used to attempt to consider meaning when dealing with natural language texts.

2.3.1. Machine Learning

Machine learning is a subfield of artificial intelligence that deals with algorithms that improve their performance by "learning" from training data. In the recent years, with the rise of deep learning the field of machine learning has advanced immensely. Neural networks have become the mainstream approach for a wide variety of tasks ranging from image recognition to price forecasting to natural language processing.

In NLP, neural networks are used for speech recognition and generation, machine translation, text classification, named entity recognition, text generation, and many other tasks. In virtual assistants and dialogue systems, neural networks are used for either end-to-end system training or, in different parts of the system, for intent detection, generation of the response text, and tracking of the dialogue state.

2.3.2. Word embeddings

A relatively recent advance in NLP was the development of methods related to word embeddings. Machine learning techniques such as neural networks normally take as an input vectors of real numbers. Therefore, for textual data to be processed by a machine learning algorithm it first has to be transformed into a vectorized form. A simple approach would be using a vector space which has as many dimensions as there are distinct words in a dictionary (typically at least 10'000-100'000). A word is then represented with a one-hot vector encoding which has value 1 in the position corresponding to that word and 0 in all others.

With word embeddings, instead of such sparse representation, the words are embedded in a less-dimensional (typically with ~100-1000 dimensions) continuous vector-space. Separate words and concepts correspond to vectors in this vector space. In this representation similar words tend to correspond to vectors that are close. The word embeddings are trained in an unsupervised manner on

large monolingual text corpora. In this way the knowledge about the language and the world is contained in word embeddings in a distilled form. There are several tools for training and using word embeddings, such as word2vec (Mikolov, Chen, Corrado, & Dean, 2013), GloVe (Pennington, Socher, & Manning, 2014) and others. One of the most recent and most successful models is fastText (Bojanowski, Grave, Joulin, & Mikolov, 2017), developed and open-sourced by Facebook AI Research. It combines several techniques to improve the quality of the representation, most importantly dividing the word into subword units.

2.3.3. Language models

Language models are probabilistic models that predict the probability of a word or sequence of words given the preceding (or surrounding) words. The NLP methods have moved from statistical language models (such as, using n-grams) to neural language models. A recent breakthrough in NLP is the advent of large-scale extensively pretrained language models, which are typically based on a deep neural network (specifically, transformer) architecture. Like word embeddings, they are trained unsupervisedly on large monolingual text corpora. In the last two years several large language models have been published, each of them further improving the state-of-the-art results on several datasets of different NLP tasks and in some of them already exceeding the human performance. Most notable examples of such models include GPT-2 (Generative Pretrained Transformer) (Radford, et al., 2019), ELMo (Embeddings from Language Models) (Peters, u.c., 2018), ULMFiT (Universal Language Model Fine-Tuning) (Howard & Ruder, 2018), BERT (Bidirectional Encoder Representations from Transformers) (Devlin, Chang, Lee, & Toutanova, 2018), Albert (A Lite BERT) (Lan, u.c., 2019).

2.3.4. NLP applications

NLP methods have a broad set of applications, including text generation, speech recognition, machine translation, sentiment analysis and many others. We focus on two tasks in the natural language understanding subfield – intent detection and parallel data extraction.

Intent detection

Virtual assistants are becoming more and more ubiquitous. On mobile phones users can interact with Amazon's Alexa, Apple's Siri, Google Assistant, Microsoft Cortana or others. An increasing number of businesses relieve the workload of their support service employees with the use of chatbots.

The first essential requirement for a successful computer-human interaction is understanding user's intent. Therefore, intent detection is one of the main tasks of a virtual assistant. The intent detection task is typically formulated in the following setting: There are several possible predefined intents according to the dialogue system domain and scope and the system should determine which one is the most relevant to the user's input. It can be solved by manually creating a list of patterns and comparing if the user's input matches any pattern. However, this method is relatively limited and the latest approach is to use machine learning methods based on neural networks trained on labelled example utterances[1]. In its most basic form, intent detection is a classification task (i.e., a supervised learning problem).

The specifics of dialog systems are that generally only small amounts of training data are available and the utterances are relatively short. It is further complicated by the specific nature of chat language, such as poorly-structured sentences, the presence of grammatical errors, the usage of informal slang, abbreviations, etc. The intent detection task is typically carried out by sequentially preprocessing, vectorizing and classifying the utterance. In the preprocessing step one or more of the following actions are performed:

[1] In spoken language analysis, an utterance is the smallest unit of speech. It is a continuous piece of speech beginning and ending with a clear pause. In intent detection context an utterance is generally a sentence or a question.

- *tokenization* - splitting the utterance into tokens, properly separating words, punctuation marks, numbers, email addresses, links, etc.,
- *automatic error-correction* - the user grammatical errors are attempted to be programmatically corrected,
- *truecasing* or *lowercasing* -- the text is converted into lowercase or true case (e.g., “usa” → “USA”),
- removal of *punctuation* marks and other symbols,
- *lemmatization* -- words are transformed into canonical form,
- *removal of stopwords* - insignificant words (such as a, and, I, or, to, etc.) are removed, either from a predefined stoplist or from a dynamically-generated list based on the training data.

During vectorization, the utterance is transformed from a textual form to a vector so it can be input to a machine learning algorithm. Most modern approaches use word embeddings (either learned from the training data, or using pretrained word embeddings from tools such as word2vec or fastText, or from a pretrained language models, such as BERT) for vectorization.

During classification the vectorized utterances are classified, typically using some machine learning algorithm such as some kind of neural network. The input for the classifier is the vectorized representation of the utterance and the output is a probability distribution over the possible intents.

Parallel corpora extraction

Parallel corpora extraction entails processing a large body of text to extract source and target language sentence pairs that match in meaning – a parallel corpus. The body of text can be a collection of documents among which some sort of correspondence for documents in source and target language exists or can be inferred – in this case the sentence pairs are extracted only looking at the matched documents. Another option is that no such correspondence exists and extraction is done on two large monolingual datasets in the respective languages. In this case, generally, a many-to-many lookup must be performed by searching the entirety of the datasets to try to align any sentences having matching meaning.

The later setting, i.e., extracting parallel sentences from two large opaque datasets, is computationally challenging – datasets that are acquired via crawling the web can measure in billions of sentences so a direct comparison is often infeasible. To alleviate this, different indirect approaches are employed, e.g, embedding the sentences of corpora in both languages into a common vector space and then extracting sentences that are close together by, e.g., doing nearest neighbour search (Artetxe & Schwenk, 2019; Thompson & Koehn, 2019; Schwenk, Wenzek, Edunov, Grave, & Joulin, 2019; Feng, Yang, Cer, Arivazhagan, & Wang, 2020).

Parallel corpora are essential to many downstream NLP tasks, such as Machine Translation and bilingual dictionary mining, where these datasets are used in Machine Learning procedures as training data.

2.4. Quantum NLP

2.4.1. Introduction to DisCoCat

We can now discuss the defined theoretical benefit of NLP on quantum devices. Among the published works in this area of particular interest are the advances of Prof. Coecke (Oxford Group and now Cambridge Quantum Computing), suggesting an approach that combines both a distributional and a compositional model of semantics using category theoretic arguments (DisCoCat) (Coecke, Sadrzadeh, & Clark, 2010). While the most successful classical NLP models are distributional models, i.e., the meaning of sentences are calculated counting the occurrence of words and its surroundings, the grammatical structure is lost. The DisCoCat model proves that the structure of entanglement in quantum mechanics is similar to that of a pregroup grammar through compact closed category theory (Lambek, 2006). In that sense, the parts of speech are mapped to the category containing finite-

dimensional vector spaces and we can make an analogy to the Hilbert space where quantum states live. Words are then composed using tensor products to make sentences, in the same way qubits can be tensorized to form many-body quantum systems.

In this model, words can be viewed as tensors that contract certain indices according to certain pregroup rules that represent grammatical structure. The outcome sentences they produce following this scheme are always 1-index tensors after the contractions, so all possible sentences live in the same vector space despite their length. This makes it possible to compare the meaning of sentences with different structures or to train a parametrized circuit to classify sentences as true/false (Meichanetzidis, Toumi, Felice, & Coecke, 2020) as well as by topic, encoding the meaning of words as unitary tunable operators acting on qubits.

One of the issues of this method is how to correctly encode words starting from a corpus. Solutions could be selecting a basis of vectors as the most common words or using amplitude encoding, but the high-dimensionality often found in NLP tasks can be problematic for current quantum devices. For that reason, tensor networks are proposed to handle the encoding of words, and dimensionality reduction from classical NLP should also be considered.

Apart from the DisCoCat model, the possible application of quantum computing to the latest NLP models such as pretrained vector embeddings, recurrent neural networks (RNNs), especially Long short-term memory (LSTM), the Transformer and other attention models, could be studied in the development of the project. Their applicability has a currently unexplored use in hybrid classical-quantum NLP problems, and may offer potential benefits due to the existing works exploring their mapping to quantum architectures.

2.4.2. Hybrid QNLP workflow

Here we focus on the hybrid approach of classical-quantum NLP tasks. Previous work in this area carried out by the ICHEC group involved a compression of the available basis states required to represent sentences of a given structure (O'Riordan, Doyle, Baruffa, & Kannan, 2020). This allowed for a corpus to be represented using a DisCoCat-like formalism, wherein sentence similarities were determined not using tensorial representation, but a Hamming-index approach across all encodings in the Hilbert space, with amplitudes weighted via a post-selection procedure.

While this model provides a novel method for encoding and defining states to represent corpus tokens, demonstrates accurate predictions of similarity, and overcomes practical implementation issues in the original DisCoCat formalism (e.g. the QRAM problem), there are some caveats. Given the preprocessing step of this algorithm requires the solution of the Hamiltonian cycle (NP-complete, though a heuristic solution is sufficient) problem to define the token orderings on a classical device, the scalability of the method is questionable for corpora requiring more than a few basis tokens to represent each meaning space. In addition, the resulting quantum circuit depths place this method into a region where NISQ devices are currently unable to accurately represent.

As such, to overcome these issues, we can plan to explore methods that map well to NISQ devices. The previously discussed variational algorithms, as well as the tensor network methods that offer compression of the data can be explored as viable routes to a solution for representing and evaluating NLP tasks. Some preliminary work has already been demonstrated in this area (Meichanetzidis, et al., 2020) (Coecke, Felice, Meichanetzidis, & Toumi, 2020), and shows effective evaluation on NISQ devices.

2.4.3. VT-NLP: Variational tensorial NLP

Following the effectiveness of variational algorithms for NISQ devices, we intend to use methods for state preparation and encoding of corpus data, along a similar line to that of (Coecke, Felice, Meichanetzidis, & Toumi, 2020) (Meichanetzidis, Toumi, Felice, & Coecke, 2020). Given the tensor-network-like relationships for describing sentence relationships, we can aim to take advantage of this formalism by representing the encoding quantum corpus state as a matrix-product state (MPS), with

operations performed to evaluate transforms and contractions using matrix product operators (MPO), as discussed in (Biamonte & Bergholm, 2017) (Orús, 2014) (Huggins, Patil, Mitchell, Whaley, & Stoudenmire, 2019) (Bai, Yang, & Chiribella, 2020).

Operations on the tensor network can be optimised to run well on both classical systems for verification, analysis and comparison of the methods. For this, we also make effective use of the variational algorithms that have been offering great promise for NISQ devices, given their tolerance towards noise. Through optimisation of tensor network bond dimension, we can explore the effect of noise on these NLP models, and better utilise the available resources of near-term quantum devices. As such, by exploiting the state preparation capabilities of variational models, and with the representability of tensor network models, we expect one can prepare states to offer a large area of exploration and data representation methods, for NLP and beyond.

We follow in Section 3 with an architectural model of the proposed software to realise this solution.

3. Proposed solution

3.1. Software architecture

In this section we present the proposed software architecture. The package will be developed in Python. The system's structure, its relation with external elements and its containers and components are introduced. We try to offer a high-level description of the software, depicting the package organization and its expected functionality, but leaving more technical details for further consideration. The rapid evaluation of models and generation of results are of the utmost importance during research and development, and so we have opted for the C4 architecture design model (Brown, s.f.) to best allow the expression of the required functionality and operations with this NLP toolkit.

In Figure 1 we show the system diagram, where our software is represented together with its environment. Quantum simulations will be carried out in the ATOS QLM, but running algorithms on physical quantum devices will also be a possibility.

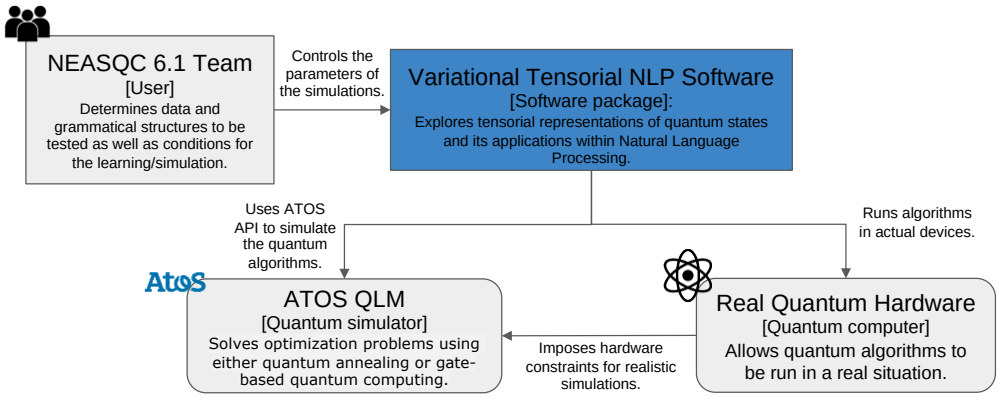

Figure 1. System-level software architecture diagram for quantum NLP solution.

Next, Figure 2 shows the containers within the software. These modules include data preparation, classical NLP, quantum computing, software testing, and benchmarking.

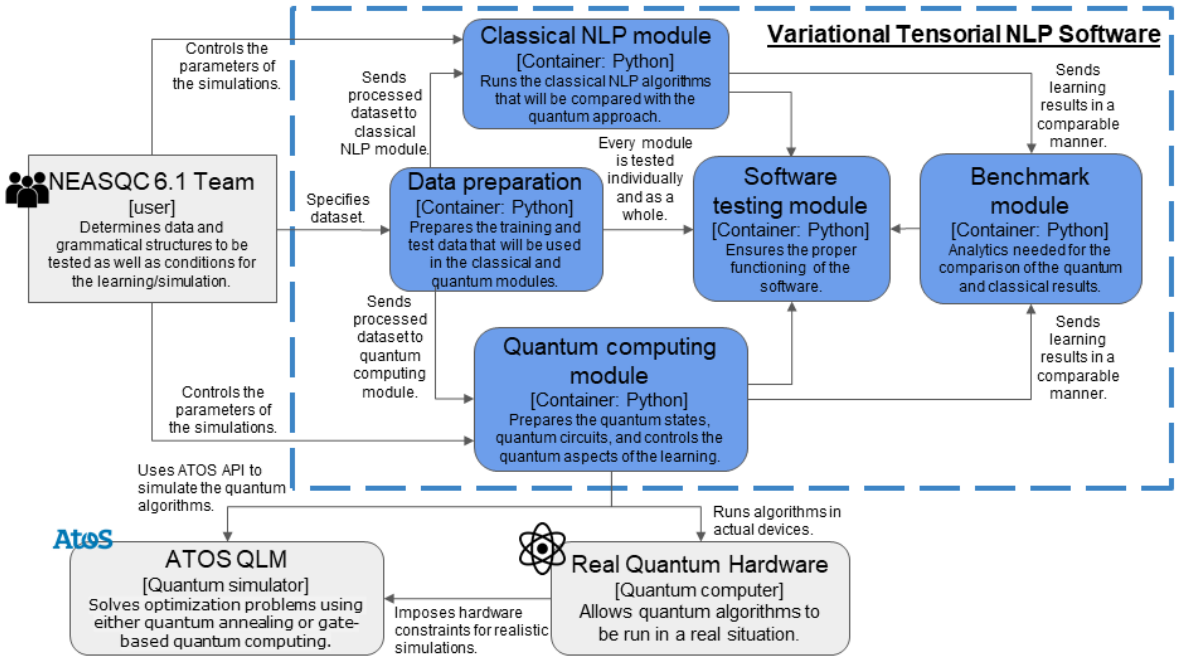

Figure 2. Container-level diagram of VT-NLP package.

Lastly, in Figures 3-7 we depict the components inside every container and how they are related to the rest of the architecture.

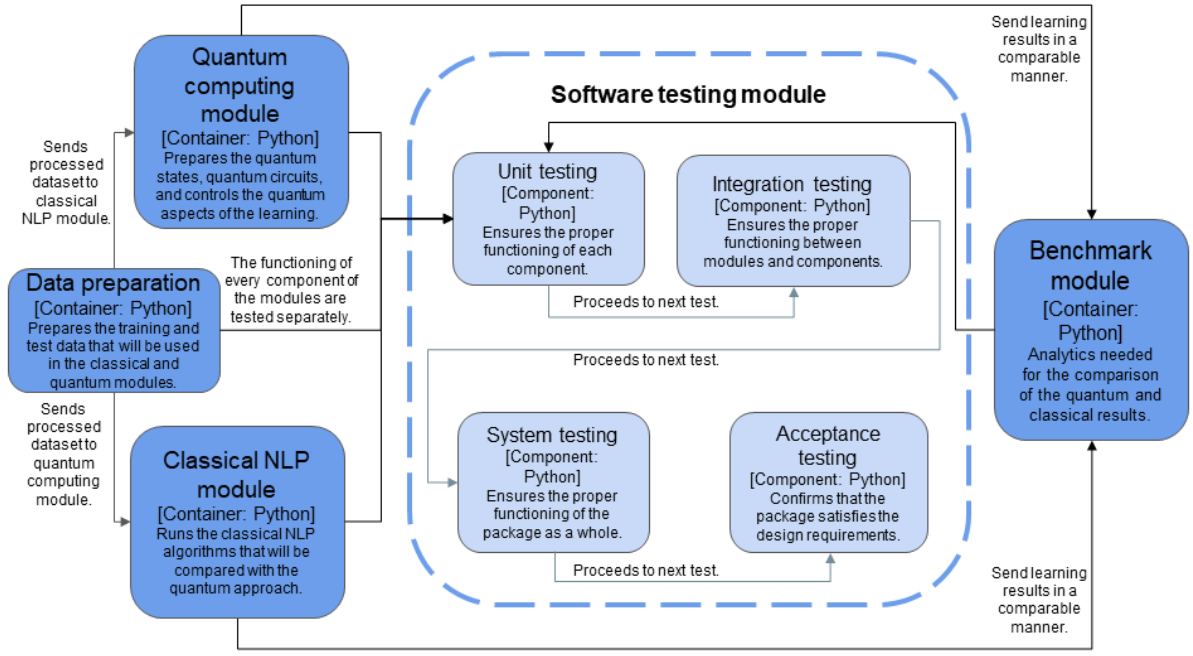

Figure 3. Component-level diagram of software testing module

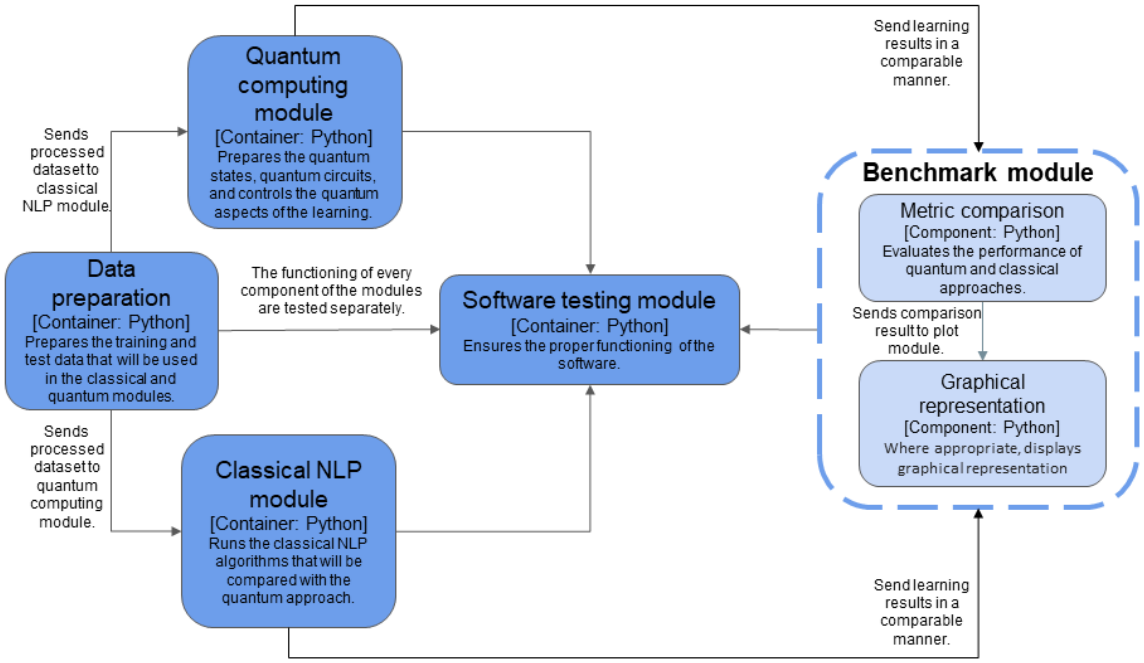

Figure 4. Component-level diagram of benchmarking module.

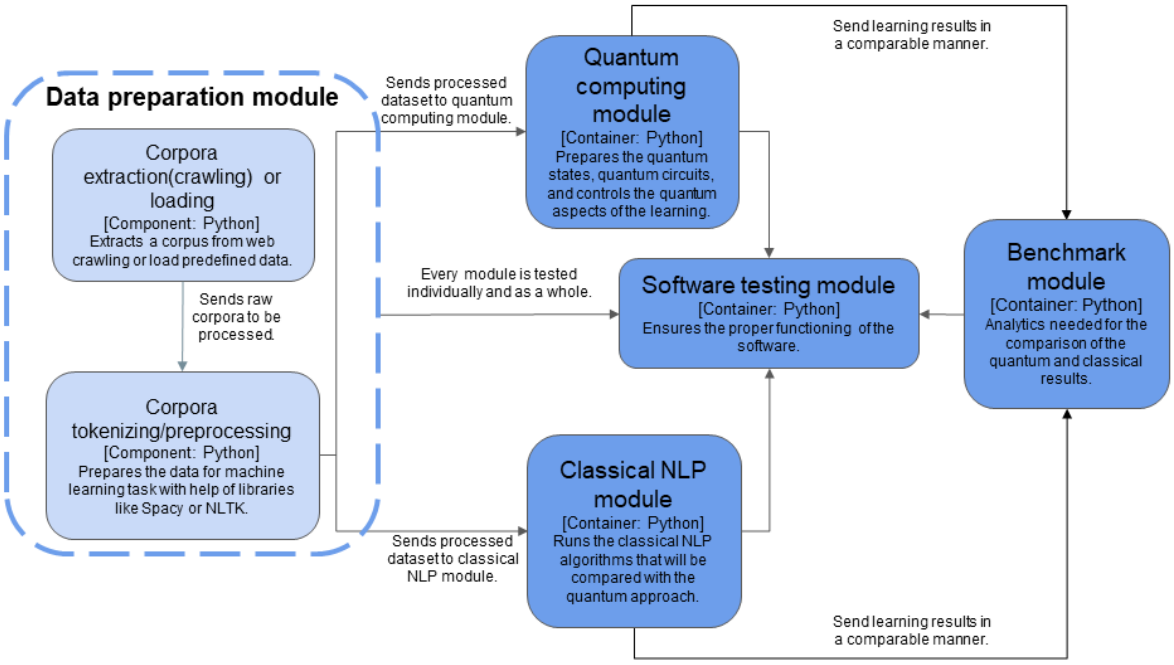

Figure 5. Component-level diagram of data preparation module.

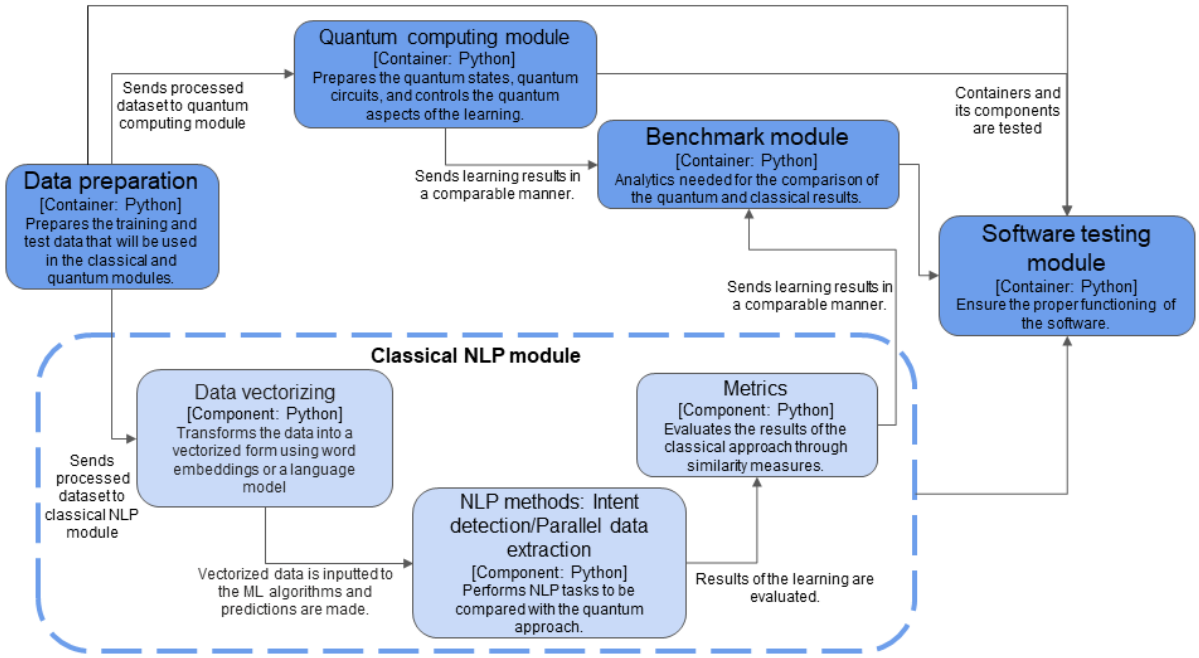

Figure 6. Component-level diagram of classical NLP module.

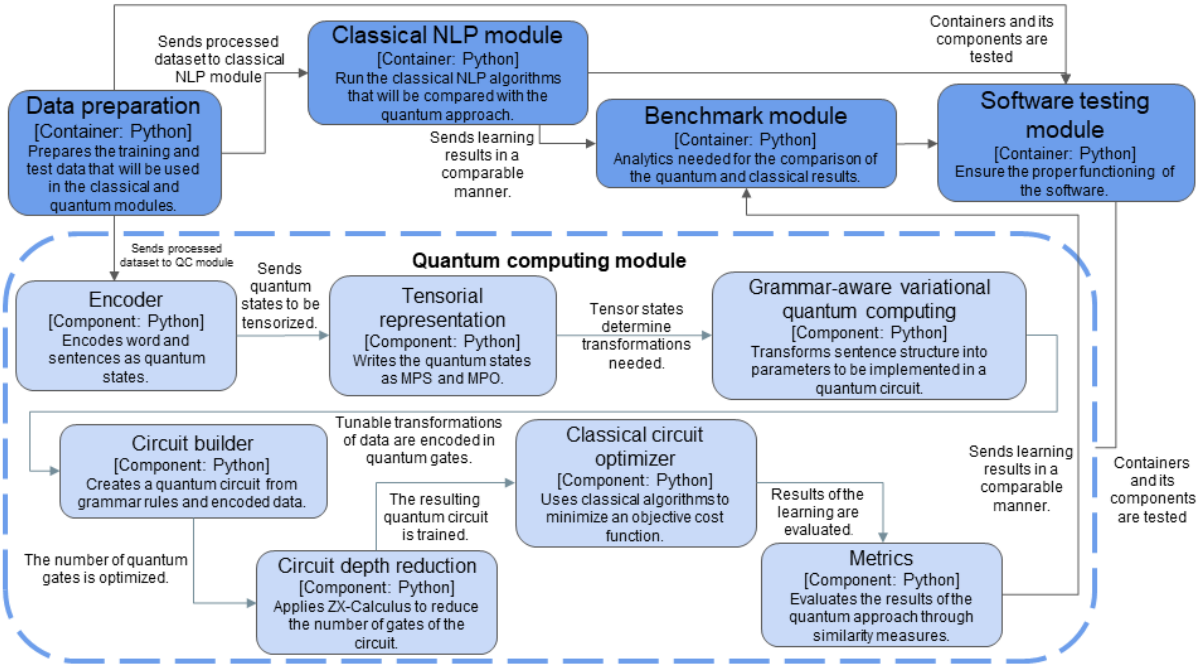

Figure 7. Component-level diagram of quantum NLP module.

Given the modular-design of the above components, we expect this software tooling will allow ease of use, ease of extensibility, and ease of integration with the myQLM platform. Each individual module will take the form of a package, with sub-modules and classes defined by the individual component functionalities. Following suit with the C4 design philosophy, we have omitted implementation specific details, as the above interfaces allow for the required expressiveness for our models.

3.2. Testing and verification

3.2.1. Software testing

The functional correctness of the classical implementation of the proposed solution will be ensured with the use of unit tests. Where appropriate, additional integration, system and acceptance tests will be performed as well. The quantum implementation will be compared to the classical implementation to evaluate the correctness and the effects of noise.

3.2.2. Benchmarking

To evaluate the performance of the methods developed within the project, the final software libraries will be evaluated on two NLP tasks - intent detection and parallel data extraction. Task-specific datasets will be sourced and the performance of the libraries will be measured through metrics specific to each task. The results will be compared to the performance of a set of baseline classical implementations (at least one for each task), which will be selected to realistically depict the general level of performance attainable by using classical NLP methods at the time of evaluation.

The specifics of the software libraries and any possible limitations of the methods are still subject to research, therefore evaluation datasets would possibly need to be adjusted to suit the limitations imposed by the libraries. One such limitation could be type of sentence structures supported by the libraries, in this case a dataset would be filtered to include only such sentences. Should other such limitations manifest, the datasets and the baseline classical implementations would be adjusted accordingly.

Intent detection

We plan to test the intent detection method on some existing intent detection dataset, used in academic evaluations, such as (Braun, Mendez, Matthes, & Langen, 2017). However, a simplified dataset with a limited sentence structure may have to be created if the limitations of the developed methods will demand it. As the intent detection datasets are typically much smaller than the ones used in parallel data extraction, creating a specific dataset from scratch might be a better solution than filtering it for suitable sentences.

The developed quantum method will be compared to the classical intent detection system developed by Tilde (Balodis & Deksne, 2019). The intent detection accuracy and F1-score will be calculated and compared for the evaluated systems.

Parallel data extraction

The developed parallel data extraction methods will be compared to the LASER (Artetxe & Schwenk, 2019) framework or any other similarly or better performing parallel data extraction method available at the time of evaluation. The selected baseline parallel data extraction method is also subject to change if any limitations appear in the methods developed in the project, that affect the design of the evaluation methodology.

Two types of evaluation will be performed – 1) a random subset of the extracted parallel data will be selected for human evaluation and 2) the whole of the extracted data will be used for extrinsic evaluation via training a neural machine-translation (NMT) system on the extracted datasets.

In the second case, the NMT systems' performance will serve as a proxy for evaluating the quality of the extracted data. The NMT systems will be evaluated using automatic metrics that are standard in NMT system evaluation, such as BLEU, CharF, BEER and/or others (Papineni, Roukos, Ward, & Zhu, 2002) (Popović, 2015) (Stanojević & Sima'an, 2014). Additionally, human evaluation will be performed to manually asses the quality of the system. The transformer model (Vaswani, et al., 2017) architecture or a similarly or better performing model available at the time of evaluation will be used as the implementation for the NMT systems.

In case the libraries don't scale to the size of corpora needed for training an NMT system, the parallel data selection methods to be evaluated will be used to extract a smaller domain-specific dataset instead. The domain-specific dataset would be used together with a larger general domain dataset that would be present in all systems' training data. The trained systems would then be evaluated on a domain-specific testing dataset.

Although LASER and similar parallel data extraction methods use bilingual sentence embeddings for similar sentence extraction across languages, using bilingual embeddings is not a hard requirement – to enable similarity comparison for sentences in multiple languages using the developed libraries, machine translation can be used to translate monolingual datasets so that similarity comparison is performed on sentences in a single language, e.g., English.

4. Acronyms and Abbreviations

Term	Definition
AI	Artificial intelligence
DisCoCat	Distributional compositional category
DMRG	Density matrix renormalisation group
ML	Machine learning
MPO	Matrix product operator
MPS	Matrix product state
NISQ	Noisy intermediate-scale quantum
NLP	Natural language processing
NLU	Natural language understanding
NMT	Neural machine translation
PQC	Parameterized quantum circuit
QAOA	Quantum approximation optimisation algorithm
QML	Quantum machine learning
QNLP	Quantum natural language processing
QRAM	Quantum random access memory
RNN	Recurrent neural network
SVD	Singular value decomposition
TN	Tensor network
VQE	Variational quantum eigensolver

Table 1: Acronyms and Abbreviations

5. List of Figures

6. List of Tables

7. Bibliography

Östlund, S. a. ((1995)). Östlund, Stellan, and Stefan Rommer. Thermodynamic limit of density matrix renormalization. *Physical review letters 75.19* .

A. Cichocki, N. L.-H. (2016). Low-Rank Tensor Networks for Dimensionality Reduction and Large-Scale Optimization Problems: Perspectives and Challenges. arXiv:1609.00893.

Andrey Kardashin, A. U. (2018). Quantum Machine Learning Tensor Network States. *arXiv:1804.02398*.

Artetxe, M., & Schwenk, H. (2019). Margin-based Parallel Corpus Mining with Multilingual Sentence Embeddings. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (lpp. 3197-3203).

Artetxe, M., & Schwenk, H. (2019). Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond. *Transactions of the Association for Computational Linguistics*, 597-610.

Bai, G., Yang, Y., & Chiribella, G. (2020). Quantum compression of tensor network states. *New Journal of Physics*, 43015.

Balodis, K., & Deksne, D. (2019). Fasttext-based intent detection for inflected languages. *Information, 10*(5). doi:10.3390/info10050161

Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., Alam, M. S., Ahmed, S., . . . Killoran, N. (2018). PennyLane: Automatic differentiation of hybrid quantum-classical computations. *arXiv: 1811.04968*.

Bharti, K. e. (2021). Noisy intermediate-scale quantum (NISQ) algorithms. *arXiv:2101.08448*.

Biamonte, J., & Bergholm, V. (2017). Tensor Networks in a Nutshell. *arXiv: 1708.00006*.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics, 5*, 135-146.

Braun, D., Mendez, A. H., Matthes, F., & Langen, M. (2017). Evaluating natural language understanding services for conversational question answering systems. *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, (lpp. 174-185).

Brown, S. (bez datuma). *The C4 model for visualising software architecture*. Ielādēts no https://c4model.com/

Cerezo, M., Sharma, K., Arrasmith, A., & Coles, P. J. (2020). Variational Quantum State Eigensolver. *arXiv: 2004.01372*.

Chan, G. K.-L., Keselman, A., Nakatani, N., Li, Z., & White, S. R. (2016). Matrix product operators, matrix product states, and ab initio density matrix renormalization group algorithms. *The Journal of Chemical Physics*, 14102.

Choquette, A., Paolo, A. D., Barkoutsos, P. K., Sénéchal, D., Tavernelli, I., & Blais, A. (2020). Quantum-optimal-control-inspired ansatz for variational quantum algorithms. *arXiv : 2008.01098*.

Coecke, B., Felice, G. d., Meichanetzidis, K., & Toumi, A. (2020). Foundations for Near-Term Quantum Natural Language Processing. *arXiv: 2012.03755*.

Coecke, B., Sadrzadeh, M., & Clark, S. (2010). Mathematical Foundations for a Compositional Distributional Model of Meaning. *arXiv: 1003.4394*.

Commission Européenne, 7. C. (2016, 10). *Grant Agreement.* Retrieved from https://shirocommunity.bull.com/ext/fpop/clouddbappliance/shareddocuments/GRANT/Grant%20Agreement-732051-CloudDBAppliance.pdf

Danilevsky, M., Qian, K., Aharonov, R., Katsis, Y., Kawas, B., & Sen, P. (2020). A Survey of the State of Explainable AI for Natural Language Processing. *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, (lpp. 447-459).

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Farhi, E., Goldstone, J., & Gutmann, S. (2014). A Quantum Approximate Optimization Algorithm. *arXiv: 1411.4028*.

Feng, F., Yang, Y., Cer, D., Arivazhagan, N., & Wang, W. (2020). Language-agnostic BERT sentence embedding. *arXiv preprint arXiv:2007.01852*.

Gambetta, J. M. (2020. gada 15. 09). *IBM's Roadmap For Scaling Quantum Technology*. Ielādēts no https://www.ibm.com/blogs/research/2020/09/ibm-quantum-roadmap/

Grimsley, H. R., Economou, S. E., Barnes, E., & Mayhall, N. J. (2019). An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nature Communications*, 3007.

Herasymenko, Y., & O'Brien, T. E. (2019). A diagrammatic approach to variational quantum ansatz construction. *arXiv: 1907.08157*.

Howard, J., & Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (lpp. 328-339).

Huggins, W., Patil, P., Mitchell, B., Whaley, K. B., & Stoudenmire, E. M. (2019). Towards quantum machine learning with tensor networks. *Quantum Science and Technology*, 24001.

IBM. (2020). *IBM's Roadmap For Scaling Quantum Technology*. Ielādēts no https://www.ibm.com/blogs/research/2020/09/ibm-quantum-roadmap/

IonQ. (2020). *Scaling IonQ's Quantum Computers: The Roadmap*. Ielādēts no https://ionq.com/posts/december-09-2020-scaling-quantum-computer-roadmap

Jurafsky, D., & Martin, J. H. (2017). *Speech and Language Processing (3rd ed. draft).* Prentice Hall.

Kissinger, A., & van de Wetering, J. (2020). PyZX: Large Scale Automated Diagrammatic Reasoning. *Electronic Proceedings in Theoretical Computer Science*, 229–241.

Kjaergaard, M., Schwartz, M. E., Braumüller, J., Krantz, P., Wang, J. I.-J., Gustavsson, S., & Oliver, W. D. (2020). Superconducting Qubits: Current State of Play. *Annual Review of Condensed Matter Physics*, 369-395.

Lambek, J. (2006). Pregroups and natural language processing. *The Mathematical Intelligencer*, 41-48.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *International Conference on Learning Representations.*

McArdle, S. e. (2020). Quantum computational chemistry. *Reviews of Modern Physics* , 015003.

McClean, J. R., Romero, J., Babbush, R., & Aspuru-Guzik, A. (2016). The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 23023.

Meichanetzidis, K., Gogioso, S., Felice, G. D., Chiappori, N., Toumi, A., & Coecke, B. (2020). Quantum Natural Language Processing on Near-Term Quantum Computers. *arXiv: 2005.04147*.

Meichanetzidis, K., Toumi, A., Felice, G. d., & Coecke, B. (2020). Grammar-Aware Question-Answering on Quantum Computers. *arXiv: 2012.03756*.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

O'Riordan, L. J., Doyle, M., Baruffa, F., & Kannan, V. (2020). A hybrid classical-quantum workflow for natural language processing. *Machine Learning: Science and Technology*, 15011.

Orús, R. (2014). A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 117-158.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311-318.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. *Proceedings of 2014 conference on empirical methods in natural language processing (EMNLP)*, (lpp. 1532-1543).

Penrose, R. (1971). Applications of negative dimensional tensors. *Combinatorial Mathematics and its Applications* (lpp. 221-244). Academic Press.

Peruzzo, A., McClean, J., Shadbolt, P., Yung, M.-H., Zhou, X.-Q., Love, P. J., . . . O'Brien, J. L. (2014). A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 4213.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *Proceedings of NAACL-HLT*, (lpp. 2227-2237).

Popović, M. (2015). chrF: character n-gram F-score for automatic MT evaluation. *Proceedings of the Tenth Workshop on Statistical Machine Translation*, 392-395.

Preskill, J. (2018). Quantum Computing in the NISQ era and beyond. *Quantum, 2*, 79.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language models are unsupervised multitask learners.* Retrieved from OpenAI blog: http://www.persagen.com/files/misc/radford2019language.pdf

Schollwöck, U. (2011). The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 96-192.

Schwenk, H., Wenzek, G., Edunov, S., Grave, E., & Joulin, A. (2019). Ccmatrix: Mining billions of high-quality parallel sentences on the web. *arXiv preprint arXiv:1911.04944*.

Stanojević, M., & Sima'an, K. (2014). Stanojević, Miloš; Sima'an, Khalil. *Proceedings of the Ninth Workshop on Statistical Machine Translation*, 414–419.

Thompson, B., & Koehn, P. (2019). Vecalign: Improved sentence alignment in linear time and space. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (lpp. 1342-1348).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention is All You Need. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 6000-6010.

Vicentini, F., Biella, A., Regnault, N., & Ciuti, C. (2019). Variational Neural-Network Ansatz for Steady States in Open Quantum Systems. *Physical Review Letters*, 250503.

Wall, M. L., Abernathy, M. R., & Quiroz, G. (2010). Generative machine learning with tensor networks: benchmarks on near-term quantum computers. *arXiv: 2010.03641*.

Wetering, J. v. (2020). ZX-calculus for the working quantum computer scientist. *arXiv: 2012.13966*.

White, S. R. (1993). Density-matrix algorithms for quantum renormalization groups. *Physical Review B 48.14*.