Complex-valued Neural Network-based Quantum Language Models

Zhang, Peng; Hui, Wenjie; Wang, Benyou; Zhao, Donghao; Song, Dawei; Lioma, Christina; Simonsen, Jakob Grue

Published in:
ACM Transactions on Information Systems

DOI:
10.1145/3505138

Publication date:
2022

Document version
Peer reviewed version

Document license:
Unspecified

Citation for published version (APA):
Complex-valued Neural Network based Quantum Language Models

PENG ZHANG, Tianjin University, China
WENJIE HUI, Tianjin University, China
BENYOU WANG, University of Padua, Italy
DONGHAO ZHAO, Tianjin University, China
DAWEI SONG, Beijing Institute of Technology, China
CHRISTINA LIOMA, University of Copenhagen, Denmark
JAKOB GRUE SIMONSEN, University of Copenhagen, Denmark

Language modeling is essential in Natural Language Processing and Information Retrieval related tasks. After the statistical language models, Quantum Language Model (QLM) has been proposed to unify both single words and compound terms in the same probability space without extending term space exponentially. Although QLM achieved good performance in ad hoc retrieval, it still has two major limitations: 1) QLM cannot make use of supervised information, mainly due to the iterative and non-differentiable estimation of the density matrix, which represents both queries and documents in QLM. 2) QLM assumes the exchangeability of words or word dependencies, neglecting the order or position information of words.

This article aims to generalize QLM and make it applicable to more complicated matching tasks (e.g. Question Answering) beyond ad hoc retrieval. We propose a complex-valued neural network-based QLM solution called C-NNQLM to employ an end-to-end approach to build and train density matrices in a lightweight and differentiable manner, and it can therefore make use of external well-trained word vectors and supervised labels. Furthermore, C-NNQLM adopts complex-valued word vectors whose phase vectors can directly encode the order (or position) information of words. Note that complex numbers are also essential in the quantum theory. We show that the real-valued NNQLM (R-NNQLM) is a special case of C-NNQLM.

The experimental results on the QA task show that both R-NNQLM and C-NNQLM achieve much better performance than the vanilla QLM, and C-NNQLM’s performance is on par with state-of-the-art neural network models. We also evaluate the proposed C-NNQLM on text classification and document retrieval tasks. The results on most datasets show that the C-NNQLM can outperform R-NNQLM, which demonstrates the usefulness of the complex representation for words and sentences in C-NNQLM.

CCS Concepts:
• Information systems → Novelty in information retrieval; Language models;
• Computing methodologies → Neural networks.

Additional Key Words and Phrases: Quantum theory, language model, question answering, neural network.
1 INTRODUCTION

Language models play an essential role in natural language processing (NLP), information retrieval (IR), and speech recognition. In scenarios from information retrieval that mainly target matching two objects (query and document), language models [66] have been used in the query likelihood model for many retrieval models. Specifically, each document is estimated as a uni-gram language model describing the probability distribution over the whole vocabulary. One of the reasons that IR communities adopt the uni-gram language model is that a longer context may lead to a more sparse data structure and larger memory. In n-gram language models, the gain from high-order term dependencies have generally been smaller than hoped [66], and come with higher computational costs due to exponentially increasing parameter space [20, 42].

Recently, by using the mathematical formulations of quantum theory, a Quantum Language Model (QLM), has been proposed to represent dependencies without expensive n-gram probabilities [45]. In QLM, single and compound words are encoded by an iteratively-optimized density matrix, and documents are subsequently ranked by calculating the von Neumann divergence between the density matrices of the query and each document. QLM is theoretically significant as a generalization of language models via the framework of quantum theory, but suffers from several problems:

- QLM did not leverage human-annotated labels in an end-to-end manner. In QLM, the iterative estimation of density matrices is processed separately from the calculation of von Neumann divergence. This makes joint training for the estimation and ranking in QLM infeasible.
- QLM assumes that words and word dependencies are interchangeable. Specifically, every projector (for a word or word dependency) is exchangeable to each other, thus neglecting the order or position information of words in a sentence. This limits the effectiveness of QLM in order-sensitive tasks.

To address the above limitations, we propose a complex-valued neural network based quantum language model (C-NNQLM) that works in an end-to-end manner. C-NNQLM adopts word embeddings to represent each word as this representation can encode richer semantic information than one-hot vectors in the original QLM. By treating each embedding vector as an observed state for each word, a sentence can be represented as a density matrix with proper normalization. This is a bottom-up approach to building density matrices based on the individual word vectors, making density matrices feasible to be integrated into a neural network architecture and automatically updated by back-propagation algorithms. The ranking by the VN-divergence in QLM is also updated with a light-weight and flexible trace inner product between the density matrices in C-NNQLM. Moreover, a modern deep learning approach, that is, a convolutional neural network, is additionally adopted to extract high-level feature from the joint representation of two density matrices.

The exchangeability assumption is to some extent reasonable in ad-hoc retrieval because the token-level exact relevance matching information is important in that setting. However, in some natural language processing tasks (e.g., question answering), order-sensitivity is typically useful. C-NNQLM generalizes QLM to a complex-valued approach to provide an extra phase vector to encode word order. Specifically, in the complex-valued end-to-end framework, a word is represented by a complex-valued word vector, with an amplitude vector encoding its word embedding information and a phase vector encoding its absolute positions. We posit that this approach facilitates...
word-position interaction in a quantum probability framework similar to the recent self-attention mechanism with word embedding and position embedding [50]. Note that as the set of real numbers is a proper subset of the set of complex numbers, extending a real-valued framework to a complex-valued one does not lose any generality. By using complex-valued components and encoding word order in phases, a quantum probability framework such as the one we present could build word order aware document representation, as opposed to the vanilla QLM [45] which adopts bag-word-word document representation.

In order to verify the effectiveness of our proposed model, we experiment on QA task including TREC, WIKI and Yahoo dataset and document retrieval task including MS MARCO dataset. By comparing our model with QLM, R-NNQLM-I is better than QLM on TRECQA (by 1.65% MAP and 3.43% MRR) and WIKIQA (by 6.85% MAP and 8.16% MRR), and the results confirmed that the end-to-end architecture is effective. In addition, by comparing C-NNQLM and R-NNQLM, we also found that extended embedding, density matrix and joint representation to the complex valued can further improve the results. In order to further explore whether the density matrix can bring performance improvement, we compare Fasttext and R-NNQLM-I, TextCNN and R-NNQLM-II on text classification. They are two types models, one type contains density matrix (including Fasttext and TextCNN), and the other type does not contain density matrix (R-NNQLM-I and R-NNQLM-II). We conducted experiments on 7 datasets such as MR, CR et al. The result also confirmed the validity of the density matrix. In addition, in the above two experiments, by comparing R-NNQLM and C-NNQLM, we also verified that the complex values brings position information to improve the model effect.

2 RELATED WORK
We now review related work on quantum-inspired retrieval models and complex-valued neural networks.

2.1 Quantum-inspired Models for IR and NLP
Quantum theory, a fundamental theory in Science, has been widely applied in a range of macro-world research areas such as social science [13], cognitive science [8, 9], speech recognition [4], natural language processing [6]. For IR, van Rijsbergen argued that quantum theory can unify the logical, geometric, and probabilistic IR models by its mathematical formalism. While this pioneering work is theoretically fundamental, at this stage, there is little evidence to support the effectiveness of a quantum retrieval approach.

After van Rijsbergen’s seminal work, much effort has been devoted to further extend the quantum IR’s theory or applications. However, this work still faces a tradeoff between the theoretical soundness and practical usefulness. For instance, Piwowarski et al. [35] proposed a quantum IR framework, with queries and documents represented by density matrices and subspaces in a principled manner, but the quantum IR framework does not show good performance in ad-hoc retrieval tasks. On the other hand, some quantum-like retrieval models are proposed based on the analogies between document ranking and quantum experiments (e.g. the double slit experiment in [72] and a photon polarization experiment in [67]). Those work achieved good results on ad hoc retrieval tasks, but used the quantum probabilistic space in an ad hoc way [45]. Wang et al. [55] explored the quantum interference in ad hoc retrieval tasks from a cognitive point of view.

Sordoni et al. [45] proposed a principled Quantum Language Model (QLM) to unify the representation of word and word dependency in a fixed-dimension vector space. It is a milestone since it theoretically generalized the statistical language modeling with the quantum probability theory, and meanwhile achieves good experimental performance in ad-hoc retrieval task. Furthermore, Sordoni et al. [44] proposed to learn latent concept embeddings for query expansion [44] using
quantum entropy minimization in QLM. Some variants of QLM have been proposed to model both word entanglement [61] and session evolution in session search task [24]. As discussed in Section 1, the original QLM and its variants do not appear to be able to utilize supervised information in an end-to-end manner, and disregards the order or position of words, thus limiting its practical applicability to some tasks.

Recently, with the trend of neural IR, in our previous work [69], we provide a Neural Network based Quantum-like Language Modeling (NNQLM) approach, which extends the original QLM architecture (a separate design for the document representation and ranking steps) to a neural network architecture (a joint optimization for the density matrix representation and the consequent ranking). By using this end-to-end architecture, NNQLM can make use of the training data and significantly outperform the original QLM on question-answering (QA) tasks, which demonstrates the application potential of NNQLM on textual matching tasks. However, NNQLM still neglects the order information of word tokens. We also notice that both QLM and NNQLM have not adopted the complex word representation, which has shown effective recently [2, 56].

In this paper, we propose a complex-valued neural network based quantum language modeling called C-NNQLM approach, in which the previous R-NNQLM is a special case. C-NNQLM adopts a complex word vector as its input, based on which the consequent density matrix representation and matching are built. In C-NNQLM, a complex word vector consists of an amplitude word vector encoding the traditional word embedding information, and a phase vector encoding the position information. C-NNQLM also extends the application task from the solely QA task to a text classification task. In summary, compared with the original NNQLM [69], both theoretical and practical aspects have been thoroughly enhanced, and thus all parts of the present paper differ substantially from those in [69].

2.2 Complex-valued Neural Network

Representations based on complex numbers have received attention due to their potential to improve optimization [31], better generalization characteristics [16], faster learning [2, 10, 59], and to allow for noise-robust memory mechanisms [10]. Moreover, using complex numbers can reduce the degree of freedoms in multiplication due to the ‘rotation’, thus potentially also decreasing the number of parameters that need to be considered [15].

Neural networks involving complex numbers have been studied extensively in previous work [11, 14, 19], and have recently seen a resurgence. For example, Reichert studied complex-valued embeddings with biological motivations, claiming that the phase of (the representation in polar coordinates of) a complex number can play the role of a gating mechanism in which input neurons with similar phases are add constructively, and neurons with dissimilar phases add destructively, and thus interfere with each other [38]. In signal processing, Shi et al. [41] quantitatively studied the importance of phase in automatic speech recognition by studying the effects of phase uncertainty on human perception. In image analysis, the information present in the phase may encode shapes, edges, and orientations, and is sufficient to recover the majority of the information encoded in its magnitude [32]. Arjovsky et al. [2] adopt complex-valued unitary transformation in RNNs to relieve the “gradient explosion issue”, and Wolter and Yao [60] proposed a novel complex-gated cell for of recurrent neural networks. Trabelsi et al. [47] developed a complex-valued neural network using several modern approaches such as complex-valued activation functions, convolution kernels, and batch normalization, and achieved state-of-the-art performance in audio-related tasks.

Complex-valued text representation has been employed in both Information Retrieval and Natural Language Processing. For example, Van Rijsbergen [49] and Melucci [28] discussed the potential and possible advantages of using complex numbers in Information Retrieval, followed by some preliminary investigations by Zuccon et al. [74]. Trouillon et al. [48] proposed complex...
embeddings for entities in Knowledge Graph Completion in order to better represent antisymmetric relations using the Hermitian inner product. Li et al. [25, 26], and Wang et al. [53] propose a unified framework with complex-valued word components to drive neural networks with a vector space based probability theory (i.e. quantum probability theory). More recently, quantum-inspired approaches using complex-valued representation has been applied to other various tasks such as multi-modal sarcasm/sentiment detection [27, 70, 71], conversational emotion recognition [23] and video sentiment analysis [22].

In this article, we apply the idea of complex word representation in neural-network based quantum language models, by designing a complex word representation and a complex density matrix representation for a sentence, based on which the text classification and question answering tasks can be performed. Noting that both quantum-like language models and complex-valued neural networks do not rely on a quantum computer, although they are based on quantum theory (quantum probability) to construct a dependency between words.

On the one hand, one disadvantage of complex-valued components may be that they may bring more parameters due to the extra phase vectors compared with real-valued components, resulting in a double amount of parameters. However, under the situation of the same number of parameters when complex-valued word embedding has half dimension of real word embedding, empirical evidence [53] shows that complex-valued word embedding also outperforms real-valued word embeddings. The improvement is likely due to the more complicated interactions in complex-valued matrix multiplications that involve real and imaginary parts (i.e., $(A+Bi)(C+Di) = (AC - BD) + (AD + BC)i$), compared to the real-valued counterpart (i.e., $AB = C$).

On the other hand, complex-valued components bring many advantages. Firstly, it provides more flexibility thanks to the phase terms. The phase terms can be considered as a gate to control how two real-valued amplitude vectors are added. Secondly, complex-valued components are essential in quantum probability at the fundamental level, especially for modeling interference effects. Lastly, using phase terms to encode sequential order is a natural and formally justified way to make use of the periodic nature of waves. For example, it could in principle generalize to arbitrarily long sequence without boundedness issues [56].

3 PRELIMINARIES ON QUANTUM PROBABILITY AND QUANTUM LANGUAGE MODELS

We briefly recapitulate basic facts on complex numbers, quantum probability theory, and the original quantum language model. The notations in this paper are listed in Appendix A.

3.1 Complex Numbers

Complex numbers are commonly used in Quantum physics. The traditional models for Information Retrieval, such as vector space models, are based on the field of real numbers ($\theta = 0$), but quantum models [51, 73] use vector spaces over the field of complex numbers

A complex number $z$ is an ordered pair of real numbers, written equivalently as $z = (a, b)$ or $z = a + bi$ where $a$ and $b$ are called the real and imaginary parts, respectively, and $i = \sqrt{-1}$. $z$ is rewritten as $z = re^{i\theta} = r(cos \theta + i sin \theta)$ in the polar complex plane as shown in Fig 1, where $r$ and $\theta$ are called the amplitude and phase respectively. In quantum theory, states are elements of a Hilbert space $\mathcal{H}$ (i.e., a vector space equipped with an inner product such that the space under the norm induced by the inner product is a complete metric space) over the field of complex numbers

---

1 For example, a 50-dimensional complex-valued word embedding (with a 50-dimensional amplitude embedding and 50-dimensional phase embeddings) is compared with an 100-dimensional real-valued vector; both cases have identical parameters in the embedding layer.

2 For practical reasons, some of the previous quantum inspired models do use the field, $\mathbb{R}$, of real numbers instead [45].
Fig. 1. Complex number in Cartesian plane and polar plane.

By transforming complex numbers to the form $re^{i\theta}$, with a amplitude $r$, the difference of two phase values determine their time/position difference as shown in Fig. 2.

Fig. 2. The order information generated by fixed amplitude and phase change, which is displayed on sine and cosine waves.

Using complex numbers in sinusoidal functions to encode word order was proposed by Wang et al. [56]. An intuitive example can be seen in Fig 2. Phase shifts allow order information encoded: to tie word order in the phase. As shifts allow a particular phase to precede another, which could model how a word precedes another. The complex-valued word representation, does not lose any expressive power capturing the semantic aspect of words. Its advantage is that, the relative distances between words, which is crucial for sequential nature of textual modeling, can be reflected by the phase difference between its complex-valued representation.

### 3.2 Basic Concepts of Quantum Probability

We use boldface letters like $\mathbf{u}, \mathbf{v}, \mathbf{w}, \cdots$ to denote column vectors in Hilbert space (unless otherwise stated, we assume these to be unit vectors). The conjugate transpose of column vector $\mathbf{u}$ is denoted by a row vector $\mathbf{u}^\dagger$, and we use the same notation for matrices. The inner product in $\mathcal{H}$ between two vectors $\mathbf{u}$ and $\mathbf{v}$ is denoted by $\mathbf{u}^\dagger \mathbf{v}$. A projector $\Pi_w$ onto the direction $\mathbf{u}$ is denoted as the outer product, i.e., an operator $\mathbf{u}\mathbf{u}^\dagger$.

$^3$A finite-dimensional Hilbert space $\mathcal{H}$ is used in this paper.
With a set of basis (orthonormal) states \( \{ \mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_j, \cdots \} \) in \( \mathcal{H} \), a superposition of the states in \( \mathcal{H} \) is a linear combination of the form

\[
\Psi = \sum_j r_j e^{i \theta_j} \mathbf{e}_j = \sum_j r_j (\cos \theta_j + i \sin \theta_j) \mathbf{e}_j
\]  

(1)

where \( r_j e^{i \theta_j} \) are complex numbers in polar coordinates, \( \{ r_j \}_{j=1}^n \) are non-negative real-valued amplitudes satisfying \( \sum_{j=1}^n r_j^2 = 1 \). \( \theta_j \) is the corresponding complex phase (\( \theta_j \in [-\pi, \pi] \)), and \( \mathbf{e}_j \) is a basis vector.

Technically, a density matrix \( \rho \) with an orthogonal basis of \( \mathcal{H} \) (denoted as \( \{ \Psi_i \} \)) and corresponding probability \( p_i \), is defined as:

\[
\rho = \sum_i p_i \Psi_i \Psi_i^\dagger
\]  

(2)

where \( \rho \) is symmetric and positive semidefinite, and the trace of \( \rho \) is 1. By Gleason’s Theorem [12], there is a bijective correspondence between the quantum probability measure \( \mu \) and the density matrix \( \rho \) (i.e., \( \mu(\rho) = tr(\rho u u^\dagger) \)).

### 3.3 The Quantum Language Model

We now briefly outline basic concepts in the original Quantum Language Model (QLM) by Sordoni et al. [45].

The quantum language model represents a word or a compound dependency between words by a quantum elementary event. For each single word \( w_i \), the corresponding projector \( \Pi_i = \mathbf{e}_i \mathbf{e}_i^\dagger \), where \( \mathbf{e}_i \) represents the word based on one-hot vector. The Quantum Language Model (QLM) [45], which first obtains the representation of density matrix \( \rho_q \) and \( \rho_d \) about the query \( q \) and document \( d \) through Maximum Likelihood Estimation (MLE). In quantum mechanics, the probability of calculating words and phrases can be calculated by Gleason’s Theorem [49] \( p(\Pi_i; \rho) = tr(\rho \Pi_i) \). Where \( p(\Pi_i; \rho) \) denotes the measured probability of the density matrix \( \rho \) on the quantum state \( \Pi_i \) corresponding to each word/term. The maximum likelihood function can be obtained by multiplying the probability obtained by measuring the quantum states corresponding to all words in the document:

\[
L_{p_d(\rho)} = \prod_{i=1}^M tr(\rho \Pi_i)
\]  

(3)

Then the maximum likelihood can be expressed as:

\[
\max_{\rho} \log L_{p_d(\rho)} = \sum_{i=1}^M \log tr(\rho \Pi_i)
\]  

(4)

The maximization above is solved by an iterative way in original QLM [45]. This process is non-differentiable so that it cannot be performed solely in an end-to-end neural network architecture but must rely on processing extraneously on the NN architecture. Additionally, QLM calculates the relevance score between \( \rho_q \) and \( \rho_d \) by negative Von-Neumann (VN) Divergence (i.e., \( -\Delta_{VN}(\rho_q||\rho_d) = tr(\rho_q \log \rho_d) \)). The two steps, i.e., the iterative optimization of such density matrices and calculating Von-Neumann (VN) Divergence, are sequentially conducted in a separated manner. Therefore, these two steps cannot be jointly trained given the annotated labels in a supervised tasks.

### 4 COMPLEX NEURAL NETWORK BASED QUANTUM LANGUAGE MODEL

We now describe our proposed end-to-end approach, called the Complex Neural Network based Quantum Language Model (C-NNQLM). We introduce the model in four steps. First, we design a
Table 1. Analogy between natural language and concepts in quantum theory [26]

<table>
<thead>
<tr>
<th>Components</th>
<th>Correspondence in quantum theory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sememe</td>
<td>basis one-hot vector / <strong>basis state</strong> ( { \mathbf{e}</td>
</tr>
<tr>
<td>Word</td>
<td>unit complex vector / <strong>superposition state</strong> ( { \mathbf{s}</td>
</tr>
<tr>
<td>Sentence representation</td>
<td>density matrix / <strong>mixed system</strong> ( { \rho</td>
</tr>
</tbody>
</table>

Note that this work not only exploits the quantum formalism to express novel IR models, but also draws on the conceptual analogy of quantum theory [7, 26, 28, 49]. The overview of the analogy is shown in Tab. 1. Specifically, we represent sememes, words, and sentences in a unified vector space from a bottom-up point of view: the basic ingredient of natural language (i.e., sememes) is considered as a set of basic states that forms a Hilbert space, while each word is considered as a combination of such sememes, a.k.a, 'superposition state' in quantum theory. The semantic composition from words to sentence is therefore modeled as particle mixing, resulting in ‘a mixed system’. The challenge of such a mixture comes from that the order between particles (words) is typically ignored, which will be solved in this paper by modeling words as waves with position information encoded in the phase terms.

The uncertainty in natural language is twofold. (1) The superposition of sememes for words is indeterminate. Such a superposition state may also be affected by the context when words are considered to be polysemous – it can be determined only if it is measured in a context. For example, given two basis vectors \( \mathbf{e}_1 \) for fruit and \( \mathbf{e}_2 \) for device, the polysemous word ‘apple’ can be represented by a superposed (uncertain) state of \( \mathbf{e}_1 \) (fruit) and \( \mathbf{e}_2 \) (device) (2) The semantic composition to compose words as sentence meaning is also uncertain, in a sense, there are no general context-free criteria. We argue these two intrinsic uncertainties in natural language make quantum theory a desirable framework in this scenario.

4.1 Word Representation

The original QLM [45] represents words in mutually-independent dimensions of space, which is common practice for exact matching tasks like ad-hoc retrieval. When processing more complicated tasks such as question answering where a few or no common terms will be shared between matching object pairs [21], QLM is ineffective as seen in the experimental section of this paper. Recently, using word vectors in end-to-end mechanisms have been proposed to relieve such mismatch issues. Generally, words can be encoded as low-dimensional, dense vectors (called “word vectors” or “word embeddings”), making it feasible to maintain the distance between words. Unlike mutually-independent word representations (i.e., one-hot vectors) utilized in QLM [45], word embeddings can not only directly inherit some universal linguistic word knowledge obtained from external large-scale corpora, but also be specifically adapted to current tasks.

\[ \text{Sometimes called “semantic matching” [21], as it is not very reliant on exact matching at the term level.} \]
Generally speaking, words can be encoded as vectors and all word vectors forms an embedding matrix \( E \in \mathbb{R}^{V \times D} \), where \( V \) is the length of the vocabulary and \( D \) is the number of embedding dimension. In this article, in addition to the utilization of word embeddings, we will also extend such word from real space \( \mathbb{R} \) to complex space \( \mathbb{C} \), in quantum language modeling process. Any real number is a special case of a complex number when its imaginary part equals to zero. Compared to real-valued word embeddings that solely capture the semantic information of words, we use complex-valued word embedding since it can both naturally encode word order and capture the semantic aspect of words in a unified space.

### 4.1.1 Complex Word Embedding

Formally, Wang et al. [53] define a general space (denoted as \( \mathcal{H} \)) for universal linguistic units, called ‘Semantic Hilbert Space’. Assume there exists a basis of \( \mathcal{H} \) with \( D \) basis vectors \( \{ e_k \}_{k=1}^D \), which forms a semantic space and any word will be one of the bases or a mixture of bases. \(^5\) Without loss of generality, each word \( s_j \) is a combination of all semantic bases with complex-valued weights \( \{ \lambda_{j,k} \}_{k=1}^D \) as below:

\[
    s_j = \sum_{k=1}^D \lambda_{j,k} e_k = \sum_{k=1}^D r_{j,k} e^{i\theta_{j,k}} e_k
\]

Since \( \lambda_{j,k} \in \mathbb{C} \), \( s_j \) is a complex-valued vector. The complex–valued vectors \( \lambda_j = \{ \lambda_{j,k} \}_{k=1}^D \) can be transformed in a polar plane \(^6\) with an amplitude vector \( r_j = \{ r_{j,k} \}_{k=1}^D \) and a phase vector \( \theta_j = \{ \theta_{j,k} \}_{k=1}^D \), that is, \( \lambda_{j,k} = r_{j,k} e^{i\theta_{j,k}} \) or its vector form \( \lambda_j = r_j \odot e^{i\theta_j} \). \(^7\) Real-valued numbers are special cases of complex-valued numbers with imaginary parts as zeros (phases are zeros). Thus, the real-valued word embedding, namely \( s_j = \sum_{k=1}^D r_{j,k} e_k \), is a special case of the complex word representation when all elements in its phase vector equal zero, i.e., \( \theta_j = 0 \).

In the Semantic Hilbert Space [53], a semantic space is defined on the basis of limited-count automatic (means it cannot be separated into smaller units) sememes. Any words would be a complex number weighted combination of the defined sememes. For example, the word ‘blacksmith’ is composed of sememes ‘human’, ‘occupation’, ‘metal’, and ‘industrial’. This suggested the word meaning itself is indeterminate meaning over sememes. Secondly, one cannot know the concrete meaning of some polysemous words without context; therefore some polysemous words are initially in an ‘indeterminate state’. Since each word is an ‘indeterminate state’, its mixture, i.e., a sentence, is by definition a mixed ‘indeterminate state’.

The interaction refers to the process of between-word semantic composition. From a bottom-up point of view, the meaning of linguistic items with larger granularities (e.g., phrases, or clauses, or sentences) is composed of linguistic items with smaller granularities (e.g., words). The path of semantic composition was typically thought to be related to the syntax but recently thought to be adaptively learned in a data-driven way without explicit syntax information. To do so, the model should be able to perceive word positions on which syntax structure may rely.

**Order-aware complex word embedding.** In this work, we associate word positions in the phase terms as below:

\[
    \theta_{j,k} = \text{pos} \times \omega_{j,k}
\]

\(^5\) Technically, these \( D \) basis vectors, called sememes in [53], are implemented as \( D \) one-hot vectors, i.e., \( e_k \) is a vector with \( k \)-th element being 1 and 0 elsewhere.

\(^6\) Any complex number \( \lambda \) in Cartesian format i.e., \( a + bi \) can be rewritten as \( \lambda = a + bi = re^{i\theta} \) in polar plane, where \( r = \sqrt{a^2 + b^2} \) and \( \theta = \tan^{-1} \frac{b}{a} \), \( i \) is the imaginary number (\( i^2 = -1 \)).

\(^7\) \( \odot \) refers to element-wise multiplication between two vectors.
where pos refers to the physical word position in a sentence or a document. We call \( \{ \omega_{j,k} \} \) as frequency embeddings. This is a simple yet effective parameterization to link word representation with word positions [56].

By adding pos as a multiplier in the phase, this allows us to model word positions as waves (see Fig. 2). Such a parameterization will result in two important properties as stated in [56]: 1) **boundedness** states that the parameterized representation is always bounded no matter how long the maximum position is, thanks to the rotation nature of phases in waves. 2) **Position-free offset transformation** means that any k-offset transformation is independent with absolute positions. This has been widely-used and effective as shown in many literature [50, 54]. In detail, suppose that we have a position encoding \( f : \mathbb{R} \rightarrow \mathbb{R}^N \). There exists a transformation \( g(k) \) that could transform \( f(n) \) to \( f(n + k) \), called a ‘k-offset transformation’. Such a k-offset transformation, i.e., \( g(k) \), should be position-free due to the exponential functional equations \( f(x + k) = f(x)f(k) \), or \( f(k) = f(x + k)/f(x) \). The position-free offset transformation is reasonable since the perception of linguistic items or even vision is usually translation invariant, as well as the pooling strategies for feature maps after convolution in vision. In other words, the absolute positions of linguistic items are usually not informative while the relative distance between linguistic items matters. The theoretical evidence about above statements needs to be further investigated, but the many works of literature widely used such a position-free offset transformation and empirically perform well. For instance, the backbone of modern natural language processing – Transformer architecture with sinusoidal position embeddings [50].

Word positions are in general continuous since 1) word positions are continuous in integer space \( \mathbb{N} \); 2) word positions are, in principle, continuous in real space \( \mathbb{R} \), although in practice we do not take any real numbers between two subsequent integers; this does mean a real number, as a word position, is invalid. By using the parameterization in Eq. 6, word order can be valid as any real number, although only integer values will be used in this paper. One can also refer to [56] which defined each word as many word functions over a continuous variable, i.e., word order.

By substituting \( \theta_{j,k} \) with \( \text{pos} \times \omega_{j,k} \) in Eq. (5), a word \( s_j \) when it appears in pos-th position in a document is represented as:

\[
\begin{align*}
\mathbf{s}_j(\text{pos}) &= \sum_{k=1}^{D} r_{j,k} e^{i(\text{pos}\omega_{j,k})} \mathbf{e}_k \\
&= \sum_{k=1}^{D} \left( r_k \left( \cos(\text{pos}\omega_{j,k}) + i \sin(\text{pos}\omega_{j,k}) \right) \right) \mathbf{e}_k \\
&= \mathbf{s}_j^* \quad \text{using Euler’s formula} (e^{ix} = \cos x + i \sin x)
\end{align*}
\]

\( \mathbf{s}_j(\text{pos}) \in \mathbb{C}^D \) is a D-dimensional vector representing word \( s_j \) which is also depending on its absolute position pos in a specific document.

An amplitude vector for a word, \( \{r_{j,k}\} \), which determines the intensity the distributions over all semantic bases, are similar to the typical word embedding which could be initialized by external word embeddings. While frequency vectors vectors, \( \{\omega_{j,k}\} \), are randomly initialized and can be updated during training. Words and their positions are two heterogeneous features, with a main difference that positions are sequential and ordered.

To guarantee the unitary property, one has to normalize each word vector. As opposed to Wang et al. [56], in this paper, we normalize each word embedding vector by dividing its Euclidean norm, namely \( s_j^* = s_j / ||s_j||_2 = s_j / \sqrt{\sum_{j,k} r_{j,k}^2} \), to obtain a unit state vector. The trainable parameters in the

\[8\]

For simplification, \( s_j \) are normalized vector \( s_j^* \) in the remaining of this paper.

complex-valued word embedding are \( \{ r_{j,k} \}_{j,k=1}^D \in \mathbb{R}^{|V| \times D} \) and \( \{ \omega_{j,k} \}_{j,k=1}^D \in \mathbb{R}^{|V| \times D} \), denoted as \( s_j^R \) and \( s_j^I \) respectively in Fig. 3, where \( D \) is the dimension of such complex word vectors, and \(|V|\) is the size of vocabulary.

4.2 Density Matrix Representation for Documents

We represent each sentence \( S \) as the density matrix \( \rho \), given by:

\[
\rho = \sum_{j=1}^{L} p_j s_j s_j^\dagger
\]

where \( \sum_j p_j = 1 \), \( \rho \) is symmetric, positive semidefinite, and \( \text{tr}(\rho) = 1 \). \( L \) is the number of words in the given documents. Each \( S_j \) is a pure density matrix, namely a semantic subspace spanned by the word embedding-based state vector \( s_j \). It also means the \( j \)-th word of the sentence \( \rho \). A linear combination of such subspace (i.e. \( s_j s_j^\dagger \)) is a mixed-state density matrix with respect to a certain probability distribution encapsulated in \( \{ p_j \} \). In formula 8, \( p_j (\sum_j p_j = 1) \) is the corresponding probability of the state \( s_j \) with respect to the \( j \)-th word in a given sentence. \( \{ p_j \} \) represent the weights of the words in different positions of the sentence, and it can be automatically updated by the backpropagation during training.

Note that the iterative estimation of density matrices in original QLM Sordoni et al. [44] is nondifferentiable. To solve this problem, we instead use a bottom-up way to build density matrices (in Eq. 8) in a differentiable manner. By doing so, we could train density matrices with the supervision of annotated labels instead of relying on counting \( n \)-gram terms in documents as original QLM Sordoni et al. [44] did.

We write \( s_j(\text{pos}) \) as \( s_j \) and \( \theta_{j,k}(\text{pos}) \) as \( \theta_{j,k} \) for simplicity.
An element in \( k \)-th row and \( z \)-th column elements of \( \rho \), i.e., \( \rho_{k,z} \) can be expressed as

\[
\rho_{k,z} = \sum_{j=1}^{L} r_{j,k} r_{j,z} e^{i(\theta_{j,k}-\theta_{j,z})} = \sum_{j=1}^{L} r_{j,k} r_{j,z} e^{i(\text{pos}_{j,k}-\text{pos}_{j,z})}
\]

where \( r_{j,k} \) and \( r_{j,z} \) are the amplitudes, and \( \theta_{j,k} - \theta_{j,z} \) are the phase difference between two \( e_k \) and \( e_z \). The diagonal elements \( \{ \rho_{k,k} \} \) are real-valued and \( \sum_k \rho_{k,k} = 1 \), while the non-diagonal elements \( \{ \rho_{k,z} \} \) in the cases \( k \neq z \) are able to capture the interactions between the basic states \( e_k \) and \( e_z \) depending on their phase difference. It turns out that the proposed representation based on complex-valued density matrices can not only encode the semantics information by means of word embeddings based amplitudes vectors, but also model the absolute position information and potentially capture correlations between individual dimensions of the feature space (denoted as the interaction terms between dimension \( z \) and \( k \) in Eq. 9).

To the best of our knowledge, current QA systems often directly align the embedding vector for each word, but without considering the higher-order feature interaction \(^{11}\). In this work, we implement such higher-order feature interaction by using mixtures of the semantic subspaces spanned by the embedding vectors. With such a mixture space, we will show that some useful similarity features/patterns will be derived in our neural network based architecture. We can also interpret the density matrix in Eq. 8 from the perspective of covariance matrices (please refer to Appendix B.2). The density matrix to some extent reflects the covariance of different embedding dimensions for a sentence. In other words, it represents how scattered the words (in the sentence) will be in the embedded space.

### 4.3 Joint Representation for Sentence Pair

The joint representation between a sentence pair (e.g., a question and an answer candidate), encapsulated as two density matrices \( \rho_q \) and \( \rho_a \) separately, are defined as their product:

\[
M_{qa} = \rho_q \rho_a
\]

(10)

Note that \( \rho_q \) and \( \rho_a \) are (complex-valued) density matrices. Simply, one can adopt the trace of \( M_{qa} \) as a parameter-free score to measure the similarity between a question and an answer candidate, i.e.,

\[
\text{score}(q,a) = tr(M_{qa})
\]

(11)

In Appendix C.2, we will explain that Eq. 11 is a well-defined trace inner product between two density matrices. Moreover, the joint representation could model word-wise similarities between two textual objects, in which such word-wise two-dimensional matching model has empirically proven more effective in \([17, 52]\). In next subsection, a deep learning approach i.e., convolutional neural network is adopted to extract high-level features from the joint representation of two density matrices \( M_{qa} \).

To intuitively understand \( M_{qa} \), we will examine the trace inner product \( tr(M_{qa}) \) to show how it can be used to capture interaction between a question and an answer.

One can write \( \rho_q \) and \( \rho_a \) in word level as below:

\[
\rho_q = \sum_{j_1=1}^{L_q} p_{j_1} s_{j_1} s_{j_1}^\dagger; \quad \rho_a = \sum_{j_2=1}^{L_a} p_{j_2} s_{j_2} s_{j_2}^\dagger
\]

(12)

\(^{10}\)Please refer to Eq. 18 in Appendix B.1) for more details.

\(^{11}\)you can check \([43]\) for an example of higher-order feature interaction in recommendation system.
We can decompose $\text{tr}(M_{qa})$ by using Eq. 25,

$$
\text{tr} (p_q p_a) = \sum_{j_1=1}^{L_q} \sum_{j_2=1}^{L_a} p_{j_1} p_{j_2} \left( s_{j_1}^+ s_{j_2} \right)^2 \\
= \sum_{j_1=1}^{L_q} \sum_{j_2=1}^{L_a} p_{j_1} p_{j_2} \left( \sum_k r_{j_1,k} e^{-i\theta_{j_1,k}} \times r_{j_2,k} e^{i\theta_{j_2,k}} \right)^2 \\
= \sum_{j_1=1}^{L_q} \sum_{j_2=1}^{L_a} p_{j_1} p_{j_2} \left( \sum_k r_{j_1,k} r_{j_2,k} e^{i(-\theta_{j_1,k} + \theta_{j_2,k})} \right)^2 \\
= \sum_{j_1=1}^{L_q} \sum_{j_2=1}^{L_a} p_{j_1} p_{j_2} \left( \sum_k r_{j_1,k} r_{j_2,k} e^{i(-\theta_{j_1,k} + \theta_{j_2,k})} \right)^2 
$$

(13)

where $r_{j_1,k} r_{j_2,k} e^{i(-\theta_{j_1,k} + \theta_{j_2,k})}$ denotes the interaction between two words, in which $r_{j_1,k} r_{j_2,k}$ is the semantic interaction term while $e^{i(-\theta_{j_1,k} + \theta_{j_2,k})}$ is the phase difference term to model relative position of two words in two sentences.

Especially when we let $\omega_{j_1,k} = \omega_{j_2,k} = \omega_{k}$, called ‘dim’ setting in Sec. 5, we have the interaction term $r_{j_1,k} r_{j_2,k} e^{i(-\theta_{j_1,k} + \theta_{j_2,k})} = r_{j_1,k} r_{j_2,k} e^{i(\text{pos}_{j_2} - \text{pos}_{j_1} + \omega_{k})}$ which solely depends on the relative distance between two words. We expect that the relative distance between two words in the question sentence and answer sentence can capture the order information and therefore boost QLM with an additional ability to perceive word order. To make positions in two sentences better compared, one could also normalize the position into a fixed range, i.e., from 0 to 1 (namely, 0 is the first token, 1 is the last token, the intermediate position will be a real number between 0 and 1). In practice, we ignore this normalization operation since we empirically found that it works well without normalization. We leave thorough investigations on the order perception between two sentences as future work. Therefore $\text{tr} (p_q p_a)$ can be considered as an ensemble of word-by-word interactions between questions and answers. In addition, the off-diagonal can also represent above information (please see Appendix C.1 for more details).

Note that in the original QLM [45], the matching score between a question and an answer is obtained by the negative VN divergence, which is hard to be integrated into an end-to-end approach. In this article, we adopt the trace inner product (i.e., score $(q, a) = \text{tr} (p_q p_a)$) as the matching score. In addition to the aforementioned properties of this similarity measurement, we will show that this measurement can be also integrated into a neural network structure.

### 4.4 Learning to Match Density Matrices

In this article, we propose a data-driven method to learn to match a textual pair encapsulated in two density matrices, Our method is called Complex-valued Neural Network based Quantum-like Language Models (C-NNQLM) with two architectures, C-NNQLM-I and C-NNQLM-II, as shown in Fig. 4 and Fig. 5, respectively.

C-NNQLM-I. C-NNQLM-I is a lightweight implementation of end-to-end neural quantum language model. C-NNQLM-I directly takes the diagonal elements of join representation $M_{qa}$ (denoted as $x_{\text{diag}}$) as features and train a binary classification layer for final predictions. Plus, the trace inner product $x_{\text{trace}}$ is used as an additional feature, since it has been used to calculate the similarity
Fig. 4. C-NNQLM-I. The first five layers are to obtain the single sentence representation, the sixth layer is to obtain the joint representation of a QA pair, and the softmax layer is to match the QA pair.

Fig. 5. C-NNQLM-II. The single sentence representation and the joint representation are the same as those in C-NNQLM-I, and the rest layers are to match the QA pair by the similarity patterns learned by 2D-Complex Convolutional Neural Networks.
between words or sentences [5, 6] and has been proven to be an approximation of the negative VN divergence [44].

In addition, the diagonal elements (forming $\mathbf{X}_{\text{diag}}$) of $\mathbf{M}_{qa}$ are also adopted to enrich the feature representation, since different diagonal elements may have different degrees of importance for similarity measurement and it is the distribution of latent concepts in the question answer pair. Then, the real-valued and complex-valued feature representation can be denoted as:

$$
x_{\text{feat}} = \left[ x^R_{\text{trace}}; x^R_{\text{diag}}; x^I_{\text{trace}}; x^I_{\text{diag}} \right]
$$

(14)

The trace inner product could be considered as a special case of Eq. 14. The differences are twofold: (1) Eq. 14 additionally makes use of off-diagonal elements while the trace inner product does not; (2) Eq. 14 adopts a weighted schema to sum all diagonal elements while the vanilla trace inner product takes a mean-weighted sum of all diagonal elements. Note that the trace inner product is parameter-free while Eq. 14 has some trainable parameters that distinguish the weights for each dimension of features.

C-NNQLM-II. In C-NNQLM-II as shown in Fig. 5, we use a “two-dimensional” (2D) convolution, max-pooling [17] to learn to match two documents. Comparing with C-NNQLM-I in which only the diagonal values and the trace value of the joint representation $\mathbf{M}_{qa}$ are used in the training process (see Fig 4), C-NNQLM-II is designed to extract more useful features rather than only the diagonal values in $\mathbf{M}_{qa}$. Therefore, in C-NNQLM-II, all elements in $\mathbf{M}_{qa}$ (rather than the diagonal ones) will be passed into a ‘two-dimensional’ (2D) convolution and row/column-wise max pooling components, resulting in a feature space

$$
x_{\text{feat}} = \left[ \text{row\_pooling}(\text{conv}(\mathbf{M}_{qa})); \text{column\_pooling}(\text{conv}(\mathbf{M}_{qa})) \right]
$$

(15)

For more details about the complex-valued convolution and row/column-wise pooling, one can refer to Appendix D. Note that a loss function should receive a real-valued loss for gradient descent, since a complex-valued loss cannot be optimized in typical neural networks. We, therefore, concatenate the real-part and imaginary-part feature vectors together as a double-sized real-valued vector before feeding them to the loss function.

The loss function. In both C-NNQLM-I and C-NNQLM-II, we use $x_{\text{feat}}$ as the encoded feature representation. Such feature representation will be fed to a fully-connected layer with a softmax activation, to project such feature to a label space. The whole neural network C-NNQLM-I is trained with the a negative cross entropy loss $^{13}$.

$$
\text{loss} = \text{cross\_entropy} (\text{softmax}(x_{\text{feat}}W + b), y)
$$

(16)

$W$ is the weights to project $x_{\text{feat}}$ to label space with the dimension of label numbers (i.e., 2, being positive or negative), and $b$ is the bias term.

Although one could design several separate MLPs to combine the individual ingredients of $X_{\text{feat}}$. We instead mix them up using a single MLP, since we believe that free combination of information from the feature vector might be beneficial for downstream tasks thanks to the data-driven training.

$^{12}$A real-valued counterpart in C-NNQLM-I is $x_{\text{feat}} = \left[ x_{\text{trace}}; x_{\text{diag}} \right]$ with only real-valued components.

$^{13}$Let $L = \sum_{i}^N \left[ y_i \log h (x_{\text{feat}}) + (1 - y_i) \log (1 - h (x_{\text{feat}})) \right]$, and $h (x_{\text{feat}})$ is the output logits for predicting the positive label.
5 EXPERIMENTAL EVALUATION

In this paper, we focus on answer selection tasks. This is a type of Question Answering (QA) where the aim is to match question and answer sentences, as opposed to matching whole queries and documents as in ad hoc retrieval. The main difficulty is that the two textual objects in QA tasks (i.e., question and answer) may not share too many common words [21]. QA tasks usually are addressed in a supervised way with given some labeled training samples. The goal is to predict whether an answer candidate (usually in a given answer candidate pool) can answer the given question as a binary document-pair classification. To deal with binary document-pair classification tasks, there are typically two modules: 1) a representation module which separately encodes a question or answer sentence; and 2) a matching module which handles two encoded representations together in an interaction space.

We evaluate the overall architecture of the proposed C-NNQLM in QA tasks in Section 5.1 and document retrieval task in Sec. 5.2, we evaluate the effectiveness of the core representation module (i.e., deep complex-valued density matrices) without the matching module in (single-document) text classification tasks (in Section 5.3). The code has been released 14.

5.1 Question Answering

Dataset. We use three well known QA datasets (summarized in Table 2): TRECQA, WIKIQA and YahooQA.

TRECQA is based on the Text REtrieval Conference (TREC) tracks 8–13 QA dataset [57]. Questions in TREC 8_12 are used as training set and questions in TREC 13 are used as development (84 questions) and testing (100 questions). The candidate answers contain correct answers and a pool of documents returned by TREC participating teams. The domain of TRECQA is factoid question answering. Manual judgement was produced for the entire TREC 13 set and the first 100 questions from the training set TREC 8_12, which is denoted as the train set by [57]. Because the train set is small, in this article, we adopt a train-all setting with extra automatically-labeled noisy samples of documents returned by TREC participating teams. The code has been released 14.

WIKIQA [64] is an open domain question-answering dataset released by Microsoft Research. The questions are from question-like queries of Bing query logs. Answer candidates are collected from the Wikipedia and labelled as correct or not through crowdsourcing (see Yang et al. [64] for details). The task is to select appropriate answers from a small answer candidate pool.

YahooQA [46] is an open-domain community QA dataset collected from Yahoo answers. Since it only provides the correct QA pairs without negative samples, as per standard practice, we construct negative examples for each question by selecting the top 4 answers based on the BM25 score of all candidate answers retrieved with the question as done in [46]. Following previous settings in YahooQA, we select the QA pairs containing questions and the best answers of length 5-50 tokens after removing non-alphanumeric characters. Note that each Question in YahooQA has only one correct answer while each question in TRECQA and WIKIQA may have multiple correct answers.

Evaluation measures. For QA, we use the following well known and standard evaluation measures: mean average precision (MAP) and mean reciprocal rank (MRR). For YahooQA, Precision@1 (P@1) and MRR are typically reported [46], for the following reason: In YahooQA, there is only a single correct answer among five answer candidates, therefore Precision@1 (P@1) and MRR is reasonable

14https://github.com/TJUIRLAB/TOIS_CNQLM
and efficient in a sense it does not need to traverse all negative samples once the correct one is found.

Table 2. Statistics of dataset.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>TREC-QA</th>
<th>WIKIQA</th>
<th>YahooQA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRAIN-ALL</td>
<td>DEV</td>
<td>TEST</td>
</tr>
<tr>
<td>#Question</td>
<td>1229</td>
<td>82</td>
<td>100</td>
</tr>
<tr>
<td>#QA pair</td>
<td>53417</td>
<td>1148</td>
<td>1517</td>
</tr>
<tr>
<td>%Correct</td>
<td>12.0</td>
<td>19.3</td>
<td>18.7</td>
</tr>
<tr>
<td>#word in sentences</td>
<td>18.656</td>
<td>16.446</td>
<td>17.033</td>
</tr>
</tbody>
</table>

**Baselines.** There are two groups of baselines: typical neural network approaches and quantum-inspired language representation approaches.

Typical neural network language representation approaches usually adopt Convolutional Neural Networks (CNNs) as the basic network architecture. Yu et al. [65] used CNNs to capture bigram information in a question/answer setting. Severyn and Moschitti [34] further developed this idea and capture n-gram of higher order dependency. Santos et al. propose Attentive Pooling (AP), which enables the pooling layer to be aware of the current input pair, in a way that information from the two input items can be explicitly interacted with each other [39].

Regarding quantum-inspired language representation approaches, in QLM, the QA sentences are represented by the density matrices $\rho_q$ and $\rho_a$, respectively. We instantiate two variants of QLM. One is the original version QLM-VN, which uses the negative Von-Neumann (VN) Divergence [45] as the matching function. The second variant, denoted QLM-trace, replaces the VN divergence with the trace inner product between $\rho_q$ and $\rho_a$ as the scoring function. QMWF [69] uses quantum many-body language modeling representation of sentence pairs and learns to match the pair through CNNs and product pooling. CNM [26] adopts complex-valued word embedding with quantum probability driven neural networks to extract similarity information from question/answer.

**Experimental settings.** Our methods are abbreviated as R-NNQLM and C-NNQLM, which denote the real and complex parts of our Neural Network based Quantum Language Model (NNQLM), respectively.

**R-NNQLM** is our real-valued end-to-end quantum language model. We instantiate two versions of it: (1) In R-NNQLM-I, the QA sentences are encoded in the real-valued density matrix, and with the embedding vector as the input. The real-valued density matrix $\rho$ is updated by a neural network. The inner product is represented by the real-valued density matrix $\rho_q$ and $\rho_a$, which is taken as the similarity between question and answer. (2) R-NNQLM-II computes the similarity between question and answer through CNNs over the joint representation of two real-valued density matrices $\rho_q$ and $\rho_a$.

**C-NNQLM** is our complex-valued end-to-end quantum language model. We instantiate two versions of it, C-NNQLM-I and C-NNQLM-II, which use the trace inner product and complex CNN to obtain the similarity between question and answer sentences, respectively.

We search the hyper parameters from a parameter pool, with learning rate in $\{0.01, 0.001, 0.0001\}$, batch size in $\{32, 60, 64, 75, 120\}$, filter number in $\{32, 64, 80, 100, 128\}$, filter size in $\{5, 10, 20, 40\}$.

The trainable parameters of C-NNQLM-II mainly include complex-valued word embedding (amplitude embedding and frequency embedding), a complex-valued CNN layer, and a prediction layer. The amplitude embeddings are trained by word2vec [29] on English Wikimedia. The dimensionality is 50, and the out-of-vocabulary words are randomly initialized by a uniform distribution
in the range of (-0.25, 0.25). The frequency embeddings are like typical word embedding, which aims to learn a map from a word index to a D-dimension word vector. Concretely, the frequency embeddings is initialized as $\omega_{j,k} = (1/10000)^k/D$, D is the dimension of word embedding, i.e., 50 in this paper.

Training frequency embeddings is different from traditional parameter updating because it involves sine and cosine activation functions. It was stated in [33] that training with sin and cosine activation functions will meet infinite local minima during optimization. However, many works [18, 54] empirically evidence it performs well by directly ignoring the above optimization issue. As many previous works did, we relieve the optimization issue with a well-designed initialization introduced by [50]. Besides the above, we could optimize frequency embeddings as a typical parameter update in backpropagation using gradient descent. To simulate complex-valued operations in complex neural networks, we use a pair of two real vectors as a substitute of a complex vector as [47] did; therefore optimizing the complex CNN layer and prediction layer is the same as classical optimization method for general neural networks.

Since frequency embeddings additionally introduce more parameters, we design two weight sharing schemes to associate frequencies: 1) with only words (i.e., $\theta_{j,k} = \theta_{j}$), denoted as C-NNQLM-II-Word; 2) with only dimensions (i.e., $\theta_{j,k} = \theta_{k}$), denoted as C-NNQLM-II-Dim. Namely, frequencies in C-NNQLM-II-Word are specific to individual words and shared among dimensions, while frequencies in C-NNQLM-II-Dim are specific to individual dimensions and shared among words.

### Table 3. Results on TRECQA and WIKIQA. Best performance per column is bold. '/' denotes that no performance was reported in prior literature. $\alpha$, $\beta$ indicate significant difference (i.e., the p-value for a two-sided t-test is strictly less than 0.05) over QLM and R-NNQLM-II respectively.

<table>
<thead>
<tr>
<th>Method</th>
<th>TRECQA</th>
<th>WIKIQA</th>
<th>YahooQA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAP</td>
<td>MRR</td>
<td>MAP</td>
</tr>
<tr>
<td>QLM-VN [45]</td>
<td>0.678</td>
<td>0.727</td>
<td>0.511</td>
</tr>
<tr>
<td>QLM-trace</td>
<td>0.668</td>
<td>0.728</td>
<td>0.511</td>
</tr>
<tr>
<td>R-NNQLM-I</td>
<td>0.679</td>
<td>0.753</td>
<td>0.546</td>
</tr>
<tr>
<td>R-NNQLM-II</td>
<td>0.750</td>
<td>0.812</td>
<td>0.650</td>
</tr>
<tr>
<td>C-NNQLM-I</td>
<td>$0.699^\alpha$</td>
<td>$0.821^\beta$</td>
<td>$0.686^\alpha\beta$</td>
</tr>
<tr>
<td>C-NNQLM-II</td>
<td>$0.712^\alpha$</td>
<td>$0.820^\alpha$</td>
<td>$0.671^\alpha\beta$</td>
</tr>
<tr>
<td>C-NNQLM-II-Word</td>
<td>$0.702^\alpha$</td>
<td>$0.812^\alpha$</td>
<td>$0.646^\alpha$</td>
</tr>
<tr>
<td>C-NNQLM-II-Dim</td>
<td>$0.736^\alpha$</td>
<td>$0.857^\alpha\beta$</td>
<td>$0.679^\alpha\beta$</td>
</tr>
<tr>
<td>CNN (Yu et al.) [65]</td>
<td>0.569</td>
<td>0.661</td>
<td>0.619</td>
</tr>
<tr>
<td>CNN (Severyn and Moschitti) [34][40]</td>
<td>0.671</td>
<td>0.728</td>
<td>0.666</td>
</tr>
<tr>
<td>QA-CNN [39]</td>
<td>0.715</td>
<td>0.807</td>
<td>0.670</td>
</tr>
<tr>
<td>QMWF [69]</td>
<td>0.752</td>
<td>0.814</td>
<td>$0.695$</td>
</tr>
<tr>
<td>CNM [26]</td>
<td>$0.770$</td>
<td>$0.859$</td>
<td>0.675</td>
</tr>
</tbody>
</table>

### Experimental Results

Table 3 summarizes the experimental results. The first group shows that the trace inner product based QLM (QLM-trace) is on a par with VN-divergence based QLM (QLM-VN), evidencing the rationality to replace the time-consuming VN-divergence distance with a lightweight trace inner product (see Sec. 4.3 for details).

In the second group of Table 3, both R-NNQLM-I and R-NNQLM-II outperform QLM variants, which shows the effectiveness of our end-to-end framework based on the density matrix. Especially, R-NNQLM-II can significantly improve the original QLM by 10.5% MAP and 11.59% MRR on...
TRECQA, and by 27.15% MAP and 28.09% MRR on WIKIQA. Note that the performance R-NNQLM-II is much better than R-NNQLM-I, since R-NNQLM-I only makes use of diagonal elements of $M_{qa}$ while R-NNQLM-II adopts convolutional Network on all elements with richer similarity features (see Sec. 4.2 for details).

In the third group of Table 3 with complex-valued neural networks, C-NNQLM-I is better than R-NNQLM-I on TRECQA (by 2.9% MAP and 9.0% MRR) and WIKIQA (by 23.2% MAP and 20.5% MRR), evidencing the effectiveness of the complex valued components. Consistently with the R-NNQLM, C-NNQLM-II improves the performance of C-NNQLM-I in all metrics (except for MAP in WIKIQA). This is due to the additional usage of all elements in $M_{qa}$, convolution components and position encoding. C-NNQLM-II significantly improves over R-NNQLM-II on TRECQA (5.5% MRR) and WIKIQA (5.4% MAP and 6.1% MRR).

Regarding the two additional settings called -Word and -Dim, -Word is relatively worse than the standard one (denoted as C-NNQLM-II) while -Dim is relatively better. In a sense, the frequency term determines how to trade off word information and position information, since the final word representation varies a lot with a extremely big frequency, while it will nearly be identical with a extremely small frequency. The experimental results suggest that such frequencies should be shared among words instead of among feature dimensions.

In the fourth group of Table 3, we compare R-NNQLM-II and C-NNQLM-II with the existing neural network based approaches ([65], [34], [40], [39], [69], [26]). The proposed C-NNQLM-II (especially C-NNQLM-II-Dim setting) achieves better performance than QLM and C-NNQLMs. Note that C-NNQLM-II achieves slightly worse performance than CNM in TRECQA and QMWF in WIKIQA. The reason of the weakness in TRECQA and WIKIQA may be that the dataset is relatively small with limited trainable samples, resulting in relatively unstable performance. While in the larger dataset YahooQA, C-NNQLM-II outperforms a strong baseline QA-CNN [39] by 19% P@1 and 10.7% MRR and QMWF [69] by 16.7% P@1 and 8.1% MRR.

### 5.2 Document Retrieval

**Dataset.** We use the MS-MARCO data set [30], which formulates a document ranking task based on the questions in the Q&A data set and the documents that answer the questions. There are 3.2 million documents, and the goal is to rank them based on their relevance. The relevant tags are derived from the passages marked as having answers in the QA data set, making this tag one of the largest relevant data sets ever. The format of the training set is composed of query, relevant-passage and non-relevant passage, which means that the problem in each sample corresponds to a positive example and a negative example. Because the data set is large, we only used 2.5% of it (about 1 million samples) as the training set due to the limited computing resources, while [36] used 5% of it as a subset.

**Baselines.** We provided various basic retrieval model. BM25 is a basic method to rank documents without training; LeToR [37] is the feature-based learning to rank, namely a support vector machine variant used for ranking problems called 'RankSVM'. K-NRM [62] is a kernel-based interaction-based neural ranker. Conv-KNRM is an n-gram version of K-NRM. We do not compare with BERT based methods because of limited computing resource. Our proposed R-NNQLM and C-NNQLM could also make use of BERT contextualized word embedding as a substitute of static word vectors like Word2Vec.

**Experimental Setup.** We search the hyper parameters from a parameter pool, with batch size in \{32, 60, 64, 128\}, learning rate in \{0.001, 0.0001, 0.00001\}, L2-regularization in \{0, 0.001, 0.0001\}, and number of hidden layer units in \{80, 100, 128\}. We adopt narrow convolution and max pooling in
CNN, with number of filters in \{64, 128\}, and size of filters in \{5, 10, 20, 40\}. We use pre-trained 50-dimensional vectors from word2vec in all models.\(^\text{15}\) For other settings, we use the same experimental step as in Section 5.1.

Table 4. Result on passage ranking in MS-MARCO Dataset. Results for baselines are form [36].

<table>
<thead>
<tr>
<th>Method</th>
<th>MRR@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM25</td>
<td>0.176</td>
</tr>
<tr>
<td>LeTor</td>
<td>0.195</td>
</tr>
<tr>
<td>K-NRM</td>
<td>0.210</td>
</tr>
<tr>
<td>Conv-KNRM</td>
<td>0.247</td>
</tr>
<tr>
<td>R-NNQLM-I</td>
<td>0.137</td>
</tr>
<tr>
<td>R-NNQLM-II</td>
<td>0.133</td>
</tr>
<tr>
<td>C-NNQLM-I</td>
<td>0.191</td>
</tr>
<tr>
<td>C-NNQLM-II</td>
<td>0.192</td>
</tr>
<tr>
<td>C-NNQLM-II-Word</td>
<td>0.186</td>
</tr>
<tr>
<td>C-NNQLM-II-Dim</td>
<td><strong>0.192</strong></td>
</tr>
</tbody>
</table>

The results are shown in Table 4. R-NNQLM-I and R-NNQLM-II underperform BM25 and some typical neural networks models including K-NRM and Conv-KNRM. While C-NNQLM-I and C-NNQLM-II outperform R-NNQLM-I, R-NNQLM-II, showing the improvement for using complex-valued components. The proposed NNQLM including (R-NNQLM and C-NNQLM) performs worse than K-NRM, and Conv-KNRM. The reason could be that the feature interaction between query and document in NNQLM is relatively simple, since the interaction of NNQLM (i.e., the trace inner product between query and document) is a single matrix multiplication while many SOTA retrieval models adopt complicated feature interaction between query and document, e.g., calculating token-wise attention across query tokens and document tokens [17, 62].

5.3 Text Classification

The proposed models includes a representation module and a matching module. Sec. 5.3 will discuss the performance of pure representation module in text classification tasks, as an ablation study to remove the matching module. We evaluate effectiveness of the core representation module (i.e. complex-valued density matrices) in text classification. The Fasttext and TextCNN are the original models which do not have density matrices.

**Dataset.** We use seven well known text classification datasets: MR, CR, SST, SUBJ, TREC, MPQA and AGNEWS (see Table 5). We use accuracy as evaluation measure based on fixed train/dev/test splits or cross validation.

**Baselines.** Based on the typical textual feature extractor for density matrices, we design two groups of baselines: Multiple Layer Perceptron (MLP) approaches and convolutional neural networks. One can also try other textual feature extractors, but the selection of feature extractor is not the main concern in this paper. After an embedding layer, Fasttext adopts a single fully-connected layer and TextCNN adopts convolutional neural networks for predictions.

**Experimental Setup.** The text classification task has only one prediction sentence, so we do not need a joint representation module. The difference between Fasttext and R-NNQLM-I is that Fasttext does not contain a density matrix while R-NNQLM does. Therefore, we use Fasttext as baseline of

\(^{15}\)This is available in https://github.com/zhaodongh/Quantum-QA/blob/master/embeddings/aquaint%2Bwiki.txt.gz.ndim%3D50.bin, which was also used by [68].
MLP. The difference between TextCNN and R-NNQLM-II is, again, that TextCNN does not contain a density matrix while R-NNQLM-II does. Therefore, we use TextCNN as a baseline of CNN.

Since MLP approaches usually process a batch of vectors instead of 2-dimensional data like density matrices, we have to transform density matrices into vectors, so two versions of R-NNQLM-I (-diag and -flat) were considered: -diag adopts only diagonal elements of density matrices; and -flat directly processes all elements of density matrices. For the density matrix, the diagonal element is the square of the word-embeddings that constitute the matrix (i.e., the self-interaction information of the word); the off-diagonal elements are the interaction information between words with different units in \{80,100,128\}. We adopt narrow convolution and max pooling in CNN, with number of filters in \{64,128\}, and size of filters in \{3,4,5\}. We use pre-trained 50-dimensional vectors from word2vec.

We search the hyper parameters from a parameter pool, with batch size in \{32,50,64\}, learning rate in \{0.001,0.0001,0.00001\}, L2-regularization in \{0,0.001,0.0001\}, and number of hidden layer units in \{80,100,128\}. We adopt narrow convolution and max pooling in CNN, with number of filters in \{64,128\}, and size of filters in \{3,4,5\}. We use pre-trained 50-dimensional vectors from word2vec in all models.

Table 6. Results on text classification datasets. \(\alpha\) and \(\beta\) indicates that the reported result is significantly improved compared to fasttext in the group I and R-NNQLM-II in the group II respectively. (The significance test is a two-sided t-test with p-value 0.05)

<table>
<thead>
<tr>
<th>Method</th>
<th>Model</th>
<th>CR</th>
<th>MR</th>
<th>MPQA</th>
<th>SST2</th>
<th>TREC</th>
<th>AGNEWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fasttext</td>
<td>MLP, (s \in \mathbb{R}^D)</td>
<td>0.726</td>
<td>0.752</td>
<td>0.867</td>
<td>0.773</td>
<td>0.736</td>
<td>0.867</td>
</tr>
<tr>
<td>R-NNQLM-I-diag</td>
<td>MLP(Diag((\rho))), (\rho \in \mathbb{R}^{D \times D})</td>
<td>0.629</td>
<td>0.755</td>
<td>0.866</td>
<td>0.829</td>
<td>0.757</td>
<td>0.732</td>
</tr>
<tr>
<td>R-NNQLM-I-flat</td>
<td>MLP(Flatten((\rho))), (\rho \in \mathbb{R}^{D \times D})</td>
<td>0.731</td>
<td>0.782</td>
<td>0.894</td>
<td>0.849</td>
<td>0.803</td>
<td>0.886</td>
</tr>
<tr>
<td>R-NNQLM-II</td>
<td>MLP(Flatten((\rho))), (\rho \in \mathbb{R}^{D \times D})</td>
<td>(\alpha)</td>
<td>(\beta)</td>
<td>(\gamma)</td>
<td>(\delta)</td>
<td>(\epsilon)</td>
<td>(\zeta)</td>
</tr>
<tr>
<td>Text-CNN</td>
<td>CNN, (s \in \mathbb{R}^D)</td>
<td>0.769</td>
<td>0.794</td>
<td>0.901</td>
<td>0.855</td>
<td>0.802</td>
<td>0.909</td>
</tr>
<tr>
<td>R-NNQLM-II</td>
<td>CNN((\rho)), (\rho \in \mathbb{R}^{D \times D})</td>
<td>0.762</td>
<td>0.794</td>
<td>0.901</td>
<td>0.855</td>
<td>0.802</td>
<td>0.909</td>
</tr>
<tr>
<td>C-NNQLM-I-diag</td>
<td>MLP(Diag((\rho))), (\rho \in \mathbb{R}^{D \times D})</td>
<td>0.759</td>
<td>0.781</td>
<td>0.896</td>
<td>0.863</td>
<td>0.758</td>
<td>0.887</td>
</tr>
<tr>
<td>C-NNQLM-I-flat</td>
<td>MLP(Flatten((\rho))), (\rho \in \mathbb{R}^{D \times D})</td>
<td>(\alpha)</td>
<td>(\beta)</td>
<td>(\gamma)</td>
<td>(\delta)</td>
<td>(\epsilon)</td>
<td>(\zeta)</td>
</tr>
<tr>
<td>C-NNQLM-II</td>
<td>MLP(Flatten((\rho))), (\rho \in \mathbb{R}^{D \times D})</td>
<td>(\alpha)</td>
<td>(\beta)</td>
<td>(\gamma)</td>
<td>(\delta)</td>
<td>(\epsilon)</td>
<td>(\zeta)</td>
</tr>
</tbody>
</table>

The results are shown in Table 6. In the topmost group, R-NNQLM-I-diag performs slightly worse than Fasttext; the reason may be that elements of word vectors are transformed from original well-trained word vector space to another distorted square space, for example, the diagonal element, i.e., \(\rho_{k,z} = \sum_j s_{j,k}^2\), where \(s_{j,k}\) is the k-dimensional element in the word vector of word \(s_j\). With further usage of all elements in density matrices, i.e., the -flat setting, the performance was better than the Fasttext baseline. Observing the experimental results, the results of R-NNQLM-I-flat are better than Fasttext, suggesting that the density matrix results in superior performance. C-NNQLM achieved the best performance, probably due to its richer expressive power and ability to encode word order.
Table 7. Ablation study of question answering about encoding position in complex-valued word embeddings.

<table>
<thead>
<tr>
<th>Method</th>
<th>TRECQA</th>
<th>WIKIQA</th>
<th>YahooQA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAP</td>
<td>MRR</td>
<td>MAP</td>
</tr>
<tr>
<td>C-NNQLM-II-Exc</td>
<td>0.709</td>
<td>0.817</td>
<td>0.625</td>
</tr>
<tr>
<td>C-NNQLM-II</td>
<td>0.712</td>
<td>0.820</td>
<td>0.671</td>
</tr>
</tbody>
</table>

In the bottom group, we see that the complex-valued density matrix with convolution network (including C-NNQLM-II-Word and C-NNQLM-II-Dim setting) obtains better results than architecture-I in all datasets. The reason is likely that convolution networks are more effective for feature abstraction than typical MLPs. The variants with different sharing schemas (C-NNQLM-II-Word and C-NNQLM-II-Dim setting) are on par with the standard C-NNQLM-II; this may be because such text classification tasks are not particularly sensitive to word order, unlike question answering tasks.

6 DISCUSSIONS

In this section, we seek to answer the following questions:

- **Q1**: does complex-valued NNQLM perform better than real-valued NNQLM?
- **Q2**: what are the effects of encoding position information?
- **Q3**: what is the difference between the end-to-end density matrix compared to the vanilla QLM [45]?

6.1 Comparison between Real-valued and Complex-valued Components (Q1)

As seen from Table 3 and Table 6, the complex-valued NNQLM in most cases outperforms real-valued NNQLM (see R-NNQLM-I and C-NNQLM-I, as well as R-NNQLM-II and C-NNQLM-II). Especially, C-NNQLM-I performs better than R-NNQLM-I in all QA tasks and document retrieval task, and 6 out of 7 text classification tasks. In the -II setting, C-NNQLM-II significantly outperforms R-NNQLM-II in terms of both MAP and MRR on WIKIQA and YahooQA, in terms of MRR@10 on MS MARCO and in terms of accuracy on 5 out of 7 text classification tasks. In particular, C-NNQLM-II improves 8 absolute points than R-NNQLM-II in terms of P@1 on YahooQA.

To remove the effect from the parameter scale, we deliberately used the -Word and -Dim settings that reduce additional parameters from \(O(|V|D)\) to \(O(D)\) or \(O(|V|)\) using sharing weights; thus the complex-valued density matrices only slightly increase the amount parameters, which is negligible compared to amplitude embedding with space complexity of \(O(|V|D)\). Observe that complex-valued NNQLM also consistently outperforms real-valued NNQLM in the parameter-efficient setting. Plus, the effectiveness of complex-valued NNQLM compared to real-valued NNQLM is independent to the following component after density matrices (i.e., MLP in -I setting or CNN in -II).

6.2 Benefit of Encoding Position (Q2)

To test the effectiveness of encoding position information, we designed a new setting that uses complex-valued embedding without modeling position information as the baseline of the ablation experiment (called ‘C-NNQLM-Exc’). As shown in Table 7 and 8, the performance decreases if position information are not encoded (‘C-NNQLM-Exc’); in particular, P1 in Yahoo dropped by 5.5 absolute point (from 0.671 to 0.616).

We also conduct a case study to intuitively show how question answering task benefit from order-sensitive modeling. The predicted answer from C-NNQLM-II could give higher score for
Table 8. Ablation study in text classification about encoding position in complex-valued word embeddings.

<table>
<thead>
<tr>
<th>Method</th>
<th>MR</th>
<th>CR</th>
<th>SUBJ</th>
<th>MPQA</th>
<th>SST2</th>
<th>TREC</th>
<th>AGNEWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-NNQLM-II-Exc</td>
<td>0.740</td>
<td>0.782</td>
<td>0.885</td>
<td>0.850</td>
<td>0.743</td>
<td>0.858</td>
<td>0.890</td>
</tr>
<tr>
<td>C-NNQLM-II</td>
<td>0.762</td>
<td>0.833</td>
<td>0.909</td>
<td>0.869</td>
<td>0.770</td>
<td>0.915</td>
<td>0.902</td>
</tr>
</tbody>
</table>

Table 9. Case study of predicted answers of C-NNQLM-II and R-NNQLM-II for given questions in YahooQA. All words are lower cased.

<table>
<thead>
<tr>
<th>Case</th>
<th>Question</th>
<th>MR</th>
<th>CR</th>
<th>SUBJ</th>
<th>MPQA</th>
<th>SST2</th>
<th>TREC</th>
<th>AGNEWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>How can Santa travel so far in such little time?</strong></td>
<td>0.971</td>
<td>0.088</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Answer:</em> Santa travel so far in a <strong>little time</strong> by using the time zone.*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Answer:</em> If you have the slim model you can perform the infamous swap trick and play copied games its a <strong>little trickier this time.</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><strong>How can I find out who supplies gas to my home?</strong></td>
<td>0.928</td>
<td>0.042</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Answer:</em> Hello to <strong>find out</strong> who the gas supplier is ring transco 08706081524 the property post code is required they will then give you the suppliers name and number to call hope this helps.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Answer:</em> <strong>Out</strong> of the friends <strong>find</strong> one you trust and ask them.</td>
<td>0.037</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 10. The difference between QLM, R-NNQLM and C-NNQLM in expressing information, mechanism.

<table>
<thead>
<tr>
<th>Models</th>
<th>inference time per sample</th>
<th>end-to-end</th>
<th>word order</th>
<th>embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>QLM</td>
<td>150ms</td>
<td>✗</td>
<td>✗</td>
<td>one-hot embedding</td>
</tr>
<tr>
<td>R-NNQLM</td>
<td>32ms</td>
<td>✓</td>
<td>✗</td>
<td>real-valued embeddings</td>
</tr>
<tr>
<td>C-NNQLM</td>
<td>95ms</td>
<td>✓</td>
<td>✓</td>
<td>complex-valued embeddings</td>
</tr>
</tbody>
</table>

Candidate answers in which phrases 'little time' or 'find out' occur together, while R-NNQLM-II which is based on bag-of-words modeling cannot perceive word order and therefore predicted the answer that contains each words in the phrases no matter each word in the phrases are far away from each other.

6.3 The Difference between End-to-end Density Matrix and the Vanilla one in QLM (Q3)

We see from Table 10 that while QLM represented a major breakthrough in introducing quantum probability to language models for QA and classification. It has three major issues that may limit its usefulness to ad-hoc retrieval. These are: (1) QLM is time-consuming due to iterative estimation of density matrix and eigendecomposition in VN-divergence. QLM takes longer time especially in prediction phase than neural network based methods. (2) Iterative estimation of density matrices is non-differentiable. Therefore, it cannot be trained in an end-to-end manner and utilize annotated labels; and (3) It is insensitive to word order.

R-NNQLM removes non-differentiable operations, namely replacing iterative estimation of density matrix with a bottom-up approach, and VN-divergence with trace inner product. Thus, R-NNQLM can be trained in an end-to-end fashion, and use word vectors trained from external large-scale corpora. By doing so, R-NNQLM has resolved the first two issues of QLM, but lacks word order information that is linguistically important for natural language. C-NNQLM extends R-NNQLM with complex-valued vector and complex-valued neural networks to encode word order in the feature level; thus it can resolve the third issue, namely that QLM and R-NNQLM are
Insensitive to word order information. Observe that QLM is more time-consuming than R-NNQLM and C-NNQLM; the latter can be sped up with GPUs using batch operations. C-NNQLM boosts the generalization ability of QLM and beats the performance of QLM on both textual matching tasks and text classification tasks by a clear margin.

7 CONCLUSIONS AND FUTURE WORK
We have proposed a Complex-valued Neural Network Quantum Language Model (C-NNQLM) containing a real-valued model (R-NNQLM) as a special case and applied it to Question Answering (QA), document retrieval, and text classification and document retrieval tasks. Our model is the first extension of QLM with neural network architectures, and one of the first quantum models to applied to textual matching tasks. C-NNQLM incorporates a new density matrix approach based on word embeddings, and the combination of such a per-sentence density matrix and the joint representation for sentence pairs, can be integrated into the neural network architectures for effective joint training. We posit that C-NNQLM is more in line with the usual workings of quantum mechanics and previous quantum-inspired language models, in particular as C-NNQLM constructs complex-valued densities based on complex-valued embeddings (inspired by the principle of quantum superposition) as the representation of a sentence, and adopts complex-valued CNNs to extract similarity from the joint representation of sentence pairs.

Systematic experiments on TRECQA, WIKIQA and YahooQA have demonstrated the applicability and effectiveness of NNQLM. The R-NNQLM-II achieves a significant improvement over QLM on both dataset, and outperforms a strong baseline aNMM [63] by 1.2% MAP and 1.6% MRR on TREC-QA. C-NNQLM-II can outperform R-NNQLM-II, by 1% (MRR) on WIKIQA, by 10.82% (P@1) and 4% (MRR) on YahooQA. In text classification tasks, the representation module of C-NNQLM achieved competitive performance compared to Text-CNN. In document retrieval task, C-NNQLM-II consistently outperforms R-NNQLM-II. However, both C-NNQLM-II and R-NNQLM-II do not perform better SOTA neural matching models; this shows that there is some space to improve NNQLMs, especially, more complicated matching module between query and document may be needed.

This work is a further investigation of QLM in an end-to-end paradigm. The interference inherently encapsulated in the way probability is used in quantum mechanics has not been deeply investigated in the current paper, and is left for future work.

8 ACKNOWLEDGEMENT
This work is supported in part by the state key development program of China (grant No., 2017YFE0111900), Natural Science Foundation of China (grant No.61772363), and the European Unions Horizon 2020 research and innovation program under the Marie SkodowskaCurie grant agreement No.721321.

A NOTATION
Some notations used in this article are specified described in Table 11.

B UNDERSTANDING DENSITY MATRICES
In this section, we will introduce two parts: a) how to understand each element in density matrices; b) interpret the density matrix from the perspective of covariance matrices.
Complex-valued Neural Network based Quantum Language Models

Table 11. The list of used notations

<table>
<thead>
<tr>
<th>notation</th>
<th>explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_{j,k}$</td>
<td>the complex-valued weights</td>
</tr>
<tr>
<td>$e_k$</td>
<td>the vector with k-th element being 1 and 0 elsewhere.</td>
</tr>
<tr>
<td>$r$</td>
<td>the amplitude</td>
</tr>
<tr>
<td>$\theta$</td>
<td>the phase</td>
</tr>
<tr>
<td>$\omega$</td>
<td>the frequency</td>
</tr>
<tr>
<td>$</td>
<td>V</td>
</tr>
<tr>
<td>$D$</td>
<td>the dimension of complex word vectors</td>
</tr>
<tr>
<td>$L$</td>
<td>the number of words in the given sentence, e.g. $L_q, L_a$</td>
</tr>
<tr>
<td>$\rho$</td>
<td>the representation of sentences by density matrix</td>
</tr>
<tr>
<td>$\rho^R_q$</td>
<td>the imaginary part representation of questions sentences by density matrix</td>
</tr>
<tr>
<td>$\rho^I_a$</td>
<td>the real part representation of documents sentences by density matrix</td>
</tr>
<tr>
<td>$M$</td>
<td>the joint representation of questions and documents</td>
</tr>
<tr>
<td>pos</td>
<td>the position of the word in the sentences</td>
</tr>
</tbody>
</table>

B.1 Understand elements in density matrices

Each element in a density matrix $\rho$ can be written as below:

$$
\rho = \sum_{j=1}^{L} p_j s^*_j
$$

$$
= \sum_{j=1}^{L} p_j \left( \sum_{k_1=1}^{D} r_{j,k_1} e^{i(\theta_{j,k_1})} e_{k_1} \right) \left( \sum_{k_2=1}^{D} r_{j,k_2} e^{i(\theta_{j,k_2})} e_{k_2} \right)^* 
$$

$$
= \sum_{j=1}^{L} p_j \left( \sum_{k_1=1}^{D} \sum_{k_2=1}^{D} r_{j,k_1} r_{j,k_2} e^{i(\theta_{j,k_2} - \theta_{j,k_1})} e_{k_1}^* e_{k_2}^* \right) 
$$

Note that any $e_{k_1}$ or $e_{k_2}$ are one hot vector, $e_{k_1}^* e_{k_2}^*$ will be a $s$ matrix with the $k_1$-th row and $k_2$-th column element being one and zero elsewhere. That is to say,

$$
\rho_{k_1,k_2} = \sum_{j=1}^{L} p_j r_{j,k_1} r_{j,k_2} e^{i(\theta_{j,k_1} - \theta_{j,k_2})} 
$$

B.2 Interpret the density matrix from the perspective of covariance matrix

We define a complex-valued word embedding for a word $w_j$ as: $s_j = [r_{j1}e^{i\theta_{j1}}, r_{j2}e^{i\theta_{j2}}, \ldots, r_{jD}e^{i\theta_{jD}}]^T$. Then the density matrix $\rho$ can be as follow:
\[
\rho = \sum_{j=1}^{L} p_j s_j s_j^\dagger
\]
\[
= \sum_{j=1}^{L} p_j \begin{bmatrix}
  r_{j,1} e^{i\theta_{j,1}} \\
  r_{j,2} e^{i\theta_{j,2}} \\
  \vdots \\
  r_{j,D} e^{i\theta_{j,D}}
\end{bmatrix} \begin{bmatrix}
  r_{j,1} e^{-i\theta_{j,1}} \\
  r_{j,2} e^{-i\theta_{j,2}} \\
  \vdots \\
  r_{j,D} e^{-i\theta_{j,D}}
\end{bmatrix}
\]
\[
= \begin{bmatrix}
  p_1 r_{1,1}^2 + p_2 r_{2,1}^2 + \cdots + p_L r_{L,1}^2 \\
  p_1 r_{1,2} r_{1,1} e^{i(\theta_{1,2}-\theta_{1,1})} + p_2 r_{2,2} r_{2,1} e^{i(\theta_{2,2}-\theta_{2,1})} + \cdots + p_L r_{L,2} r_{L,1} e^{i(\theta_{L,2}-\theta_{L,1})} \\
  \vdots \\
  p_1 r_{1,D} r_{1,1} e^{i(\theta_{1,D}-\theta_{1,1})} + p_2 r_{2,D} r_{2,1} e^{i(\theta_{2,D}-\theta_{2,1})} + \cdots + p_L r_{L,D} r_{L,1} e^{i(\theta_{L,D}-\theta_{L,1})}
\end{bmatrix}
\]

(19)

Secondly, we define the matrix \(X\) and \(P\) as:
\[
X = \begin{bmatrix}
  s_1, & s_2, & \cdots, & s_L
\end{bmatrix}
\]
\[
P = \begin{bmatrix}
  p_1 & 0 & \cdots & 0 \\
  0 & p_2 & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & p_L
\end{bmatrix}
\]

(20)

Next, we calculate the result of \(XPX^T\):
\[
XPX^T = 
\begin{bmatrix}
  r_{11} e^{i\theta_{11}} & r_{21} e^{i\theta_{21}} & \cdots & r_{L1} e^{i\theta_{L1}} \\
  r_{12} e^{i\theta_{12}} & r_{22} e^{i\theta_{22}} & \cdots & r_{L2} e^{i\theta_{L2}} \\
  \vdots & \vdots & \ddots & \vdots \\
  r_{1D} e^{i\theta_{1D}} & r_{2D} e^{i\theta_{2D}} & \cdots & r_{LD} e^{i\theta_{LD}}
\end{bmatrix}
\begin{bmatrix}
  p_1 e^{-i\theta_{11}} & p_2 e^{-i\theta_{12}} & \cdots & p_L e^{-i\theta_{1D}} \\
  p_1 e^{-i\theta_{21}} & p_2 e^{-i\theta_{22}} & \cdots & p_L e^{-i\theta_{2D}} \\
  \vdots & \vdots & \ddots & \vdots \\
  p_1 e^{-i\theta_{L1}} & p_2 e^{-i\theta_{L2}} & \cdots & p_L e^{-i\theta_{LD}}
\end{bmatrix}
\]
\[
= \rho
\]

(21)

Through the above formula, we can draw the conclusion:
\[
\rho = \sum_{j=1}^{L} p_j S_j = \sum_{j=1}^{L} p_j s_j s_j^\dagger = XPX^T
\]

(22)

where \(X\) is composed of \(s_1, s_2, \cdots, s_L\), \(X \in \mathbb{C}^{L \times D}\); \(P\) is composed of \(p_1, p_2, \cdots, p_L\), \(P \in \mathbb{R}^{L \times L}\), and the off-diagonal elements of \(P\) are all zeros.
We can clearly see that the different columns of the matrix are the word-embeddings representations of different words, and each line is the same dimension of different words. We not only increase the speed of code execution, but also represent each element of the density matrix the interactive information between the dimensions of the word by converting the operation of the vector outer product to the operation of matrix multiplication.

C UNDERSTANDING THE PRODUCT OF TWO DENSITY MATRICES

In order to analyze the property of this joint representation, we can first decompose the density matrix of the question through spectral decomposition:

$$\rho_q = \sum_{j=1} p_j s_j s_j^\dagger$$ (23)

Where $p_j$ is an eigenvalue and $s_j$ is the corresponding eigenvector. The eigenvector can be interpreted as a latent semantic basis, and the eigenvalue can reflect how scattered the words are in the corresponding basis. Similarly, the answer density matrix $\rho_a$ can be decomposed into $\rho_a = \sum_k p_k s_k s_k^\dagger$. Then the joint representation between $\rho_q$ and $\rho_a$ can be written as:

$$\rho_q \rho_a = \sum_{j_1,j_2=1} p_{j_1} p_{j_2} s_{j_1} s_{j_1}^\dagger s_{j_2} s_{j_2}^\dagger = \sum_{j_1,j_2=1} p_{j_1} p_{j_2} s_{j_1}^\dagger s_{j_2} s_{j_1} s_{j_2}^\dagger$$ (24)

Since $s_{j_2}^\dagger s_{j_1} = tr \left( s_{j_2} s_{j_1}^\dagger \right)$, its trace will be

$$tr \left( \rho_q \rho_a \right) = \sum_{j_1,j_2=1} p_{j_1} p_{j_2} \left( s_{j_1} s_{j_2} \right)^2$$ (25)

which is the sum of similarities in word level. Note that $w_j$ is from a question document while $w_k$ is from an answer document. Thus, the joint representation matrix $M_{qa}$ can encode the similarity information across a question and an answer candidate by ensembling their word-wise similarities. More generally, $tr \left( \rho_q \rho_a \right)$ is a generalization of inner product from vectors to matrices, which is called trace inner product [3] as introduced in the following subsection.

C.1 OFF-DIAGONAL ELEMENTS of JOINT REPRESENTATION

We have discussed the meaning of the diagonal elements of the joint representation matrix above. In this section, we will study the meaning of off-diagonal elements.

The joint representation matrix is defined as:

$$M_{qa} = \rho_q \rho_a$$

$$= \sum_{j_1,j_2=1} p_{j_1} p_{j_2} s_{j_1}^\dagger s_{j_1} s_{j_2} s_{j_2}^\dagger$$

$$= \sum_{j_1,j_2=1} p_{j_1} p_{j_2} s_{j_1}^\dagger s_{j_2} \left( \sum_{k_1} r_{j_1,k_1} e^{i\theta_{j_1,k_1}} \sum_{k_2} r_{j_2,k_2} e^{-i\theta_{j_2,k_2}} \right)$$ (26)

$$= \sum_{j_1,j_2=1} p_{j_1} p_{j_2} s_{j_1}^\dagger s_{j_2} \left( \sum_{k_1} r_{j_1,k_1} r_{j_2,k_2} e^{i\left(\theta_{j_1,k_1}-\theta_{j_2,k_2}\right)} \right)$$

$$= \sum_{j_1,j_2=1} p_{j_1} p_{j_2} s_{j_1}^\dagger s_{j_2} \left( \sum_{k_1,k_2} r_{j_1,k_1} r_{j_2,k_2} e^{i\left(\text{pos}_{j_1,k_1}-\text{pos}_{j_2,k_2}\right)} \right)$$

when $k_1 \neq k_2$, it means off-diagonal elements of joint representation. The off-diagonal elements represent the interactive information between the different dimensions of each word in the sentence. pos$_{j_1} \omega_{j_1, k_1} - \text{pos}_{j_2} \omega_{j_2, k_2}$ represents the position information interaction in different dimensions between the position of the word $s_{j_1}$ and the position of the word $s_{j_2}$ in two sentences. It also represents the relative position information of word $s_{j_1}$ and word $s_{j_2}$, which word $s_{j_1}$ belongs to question sentence and word $s_{j_2}$ belongs to answer sentence. $r_{j_1, k_1} r_{j_2, k_2}$ represents the semantic information interaction in two words. In the matching task, multiply the density matrix of two sentences, and the information of the non-diagonal elements can be regarded as the interactive information between the words in the two sentences and the different dimensions of the words, which can capture richer semantic information.

C.2 TRACE INNER PRODUCT

Claim 1. In a complex $m \times n$ space called $\mathbb{C}^{m \times n}$, a map, $\langle \cdot, \cdot \rangle : \mathbb{C}^{m \times n} \times \mathbb{C}^{m \times n} \to \mathbb{R}$, defined as

$$f(A, B) = \text{trace}(AB^\dagger)$$

is a well-defined inner product, i.e. it does satisfy conjugate symmetry, linearity and definiteness.

Proof. We will give simple proof for those three properties respectively.

Definiteness of Trace Inner Product
For any density matrix $A = (A_{ij}) \in \mathbb{R}^{m \times m}$, we have:

$$\langle A, A \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} A_{ji}^\dagger = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij}^2 \geq 0 \quad (27)$$

Conjugate Symmetry of Trace Inner Product
It’s well-known $\text{tr}(A) = \text{tr}(A^\dagger)$, then we have

$$\langle A, B \rangle = \text{tr}(AB^\dagger) = \text{tr}\left(\left(AB^\dagger\right)^\dagger\right) = \text{tr}(BA) = \langle B, A \rangle \quad (28)$$

We can clearly see that the trace inner product satisfies the conjugate symmetry property.

Linearity of Trace Inner Product
Assume there is a real number $\lambda$, since the two theory as follow:

$$\text{tr}(\lambda X) = \lambda \text{tr}(X)$$

$$\text{tr}(A + B) = \text{tr}(A) + \text{tr}(B) \quad (29)$$

Then we have:

$$\langle \lambda A + B, C \rangle = \text{tr}\left(\left(\lambda A + B\right) C^\dagger\right)$$

$$= \text{tr}\left(\lambda AC^\dagger + BC^\dagger\right)$$

$$= \lambda \text{tr}(AC^\dagger) + \text{tr}(BC^\dagger)$$

$$= \lambda \langle A, C \rangle + \langle B, C \rangle \quad (30)$$

□

D THE IMPLEMENTATION OF COMPLEX-VALUED CONVOLUTION AND POOLING

In this work, we adopt 2D complex convolution kernels to extract higher-level information from $M_{qa}$. We convolve a complex-valued matrix $M_{qa} = A + iB$ by a complex-valued matrix $W = C + iD$, where $A$ and $B$ represent $p_{r_q}^R p_{r_a}^R - p_{r_q}^R p_{r_a}^I$ and $p_{r_q}^R p_{r_a}^I + p_{r_q}^R p_{r_a}^R$, respectively, as well as $C$ and $D$ are initialized real matrices. As the convolution operator is distributive, convolving the matrix $M_{qa}$ by the filter $W$ we obtain:

\[ M_{qW} = (A \ast C - B \ast D) + i (B \ast C + A \ast D) \] (31)

Suppose the number of filters is \( o \). The \( j \)-th convolution operation is formulated as:

\[
O^R_j = \delta (A \ast C_j - B \ast D_j) \\
O^I_j = \delta (B \ast C_j + A \ast D_j)
\] (32)

where \( 1 \leq j \leq o \), \( \delta \) is the non-linear activation function, \( \ast \) denote the 2D convolution, \( C_j \) and \( D_j \) are the real and imaginary parts of the weight matrix for the \( j \)-th complex convolution kernel, respectively. \( O^R_j \) is the real part of the feature map and \( O^I_j \) is the imaginary part of the feature map.

After the convolution layer obtains the complex feature maps, we then use 2-D max-pooling to generate vectors \( r^R_j \in \mathbb{R}^k \) and \( r^I_j \in \mathbb{R}^k \), for the real part and the imaginary part of the feature map respectively, with the formulas as follows:

\[
r^R_j = (R_k : k = 1, ..., k) \\
R_j = \max \{ O^R_j \} \\
r^I_j = (I_k : k = 1, ..., k) \\
I_j = \max \{ O^I_j \}
\] (33, 34)

Then we concatenate these vectors as follows:

\[
x_{\text{feat}} = [r^R_1; r^I_1; ...; r^R_j; r^I_j; ...; r^R_c; r^I_c]
\] (35)

where \( 1 \leq j \leq c \). The \( x_{\text{feat}} \) is then passed through a fully connected hidden layer, which allows for modeling interactions between the density of questions and answers. Finally, the output of the hidden layer will be fed to the softmax layer, and it generates a distribution about the class labels.

All in all, the two-dimensional complex-valued convolution operation of C-NNQLM aims to extract useful similarity information from the joint representation. We can adjust the parameters of the kernels and hidden layer weights while training the model.[1]

REFERENCES


Complex-valued Neural Network based Quantum Language Models


1:32 Peng Zhang et al.


