UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Universitá degli Studi di Padova

Dipartimento di Ingegneria dell' Informazione

Dynamic Content Monitoring and Exploration using Vector Spaces

Dottorando: Benyou Wang

Ph.D. Thesis

30th September, 2021

Dynamic Content Monitoring and Exploration using Vector Spaces

Scuola di Dottorato di Ricerca in Ingegneria dell' Informazione
Ciclo XXXIV

Submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Benyou Wang

Direttore della scuola: Ch.mo Prof. Andrea Neviani
Supervisore: Ch.mo Prof. Massimo Melucci
Co-Supervisore: Dr. Emanuele Di Buccio

To my newly-born son, Jiawen.

...

Acknowledgement

My Ph.D. project is funded by the Quantum Information Access and Retrieval Theory (QUARTZ) project, which has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 721321. I would like to express my sincere gratitude to the QUARTZ project for accepting me into the project and supporting my Ph.D. research. This may also thank the pioneer of Quantum IR, Cornelis Joost "Keith" van Rijsbergen.

QUARTZ aims to develop a novel theoretically and empirically motivated approach to Information Access and Retrieval (IAR) based on the quantum mechanical framework. The quantum mechanical framework is expected to connect between classical ranking models and their context, especially in multimodal scenarios. I worked as the ESR 2 (Early Stage Researcher) in QUARTZ in there three years, which objective is to monitor dynamic content using vector spaces, see `http://www.quartz-itn.eu/people/esr/esr-2`. Thanks to the development of neural networks, words, the basic ingredient in IAR, are generally embedded as dense vectors in vector space, which inspires this work to first model words like particles in Hilbert space. Interestingly, beyond the analogous particle properties of words, this work also explores the wave properties to capture their dynamic aspect. Such dynamic aspects of words could be related to 1) their spatial position: the order matters since natural language is a sequence of tokens; and 2) their temporal timestamp: word meaning may change over time. Therefore, the work is divided into two parts packages: 1) modeling words as particle and 2) modeling words as waves. The former is listed in Chapter 3. The latter is about two scenarios including a spatial case and a temporal case that is introduced in Chapter 4 respectively.

I would like to thank my supervisor Massimo Melucci, the one who always set high standards for what good research means. In the very beginning, he was super patient with my bad English communication skills. Also, he repeatedly trained me to orally present technical content in front of him, which was sometimes very boring for him due to the weird accent, unreasonable logic, and immature content. In research itself, I was told to treat technical statements rigorously, cite existing work comprehensively, and read the original books. All of the above will benefit my whole life. Plus, I will always remember the motto from him: 'cold pizza is the worst pizza in the world'.

I would like to thank another supervisor Emanuele Di Buccio. He spent a lot of time supervising me by arranging weekly meetings and timely help me whenever I need. Without his help, my thesis could not be finished with the current quality. In academic training, he carefully checked my written materials including notes, reports, papers – even he read every cited paper to make everything concisely and precisely stated. Other than research, honesty is another lesson he taught me. If I became a professor, I could definitely convey these training criteria from Emanuele and Massimo to my students.

Many thanks to all my colleagues at the Department of Information Engineering, and fellow QUARTZ researchers. Qiuchi Li and Prayag Tiwari are the two guys who accompany

me in my whole Ph.D. We spent some fruitfully and joyful moments in the office, the apartment, as well as lovely lakes, mountains, and seas in Italy. Lucas hosted me in Copenhagen during my secondment. Dongsheng took me to have much fun together. I also enjoy the discussions with Sager, Dimitris, and Amit.

I would like also thank to Christina and Jacob in UCPH. They not only supervise me about what is wrong or correct but also why it is and how to deal with it. I wish I could work more with the two professors in the future. Many thanks to Jian-Yun Nie, from whom I would learn a lot about the research curiosity and ambition. I felt sorry that the collaborated project was not properly finished due to the COVID 19 and time limit.

I would like also thank to Peng Zhang and Dawei Song, who encouraged me to return to research from Tencent, and join the QUARTZ family and UNIPD. Especially thank both of them to bring me to the academic career in the master phase.

Diederik Aerts, Peter Bruza, Ingo Frommholz, Haiming Liu, Ingo Schmitt, Andrei Khrennikov gave nice suggestions and comments on the ongoing work during my Ph.D. Pierpaolo Basile and Roberto Basili made useful suggestions for the thesis.

I also want to thank Maarten and Pengjie Ren for hosting me in Amsterdam, Alexsandro and Siva Reddy hosting me in MILA, Hang Li hosting me in ByteDance, Lifeng Shang, Xin Jiang, and Qun Liu hosting me in Huawei, Shengyu Zhang hosting me in Tencent Quantum Lab, Zhaochun Ren and Pengjie Ren hosting in Shandong University, Pan Zhang hosting me in the theoretical institute of physics, Tiezheng Ge hosting me in Alibaba, Alex Meng hosting me in the Tencent.

I also thank the following friends (in random order) who I met during my Phd. They are Wei Zhao, Jiahuan Pei, Chen Zhang, Qianqian Xie, Jianquan Li, Xiaokang Liu, Liqun Ma, Zhan Su, Min Yang, Teng Ma, Chengguang Guo, Chunyan Cheng, Jietuo Wang, Yan Hu, Zhenmi Xu, Peng Lu, Haiming Wu, Jie Fu, Yikang Shen, Zhenzhen Li, Yuxin Ren, Shaobo Li, Xiaoguang Li, Zhihong shao, Chenyang lv, Lu hou, Baojun Wang, Hao Yang, Wei Zhang, Jian Li, Zhao Li, Xiaoliu Mao, Shuai Zhang, Donghao Zhao, Wenjie Hui, Ran Liu, Sunzhu Li, Ningning Wang, etc.

I thank all the lecturers and staff (including IT staff, administrative stalls like Debora, Enrico, and Cristina) in DEI and UNIPD, who gave me some help or just said hello to me.

Most importantly, my last thanks go to my family including my wife and parents, and my sister (and his lovely daughters), especially my son, the little 'best paper'.

I would like to share the meaning of my name. Benyou is actually related to two Chinese characters: 本(Ben) and 友(You). The former 本(Ben) was given by my ancestors with the surname Wang, according to the depth of family tree – all of my brothers would have the same middle name 本(Ben). The latter is given by my grandfather (a poor but self-reliant farmer). 本(Ben) in Chinese means 'the root, or the original', which was used as a verb meaning 'explore the origin of the physical world' [1]. And I want to share the meaning of my middle name 本(Ben) to the people who are pursuing knowledge: 'people are equally enjoying the knowledge regardless of their educational background and material wealth'. The wonderful adventure to pursue a Ph.D. degree in UNIPD makes me get a great sense of achievement from knowledge. I really enjoy freely exploring unknown knowledge. Hope that I would work in academia for my whole life if I could.

[1] See a sentence in 'The Seven Style' (七发) from a litterateur Mei Chang (枚乘) in the Western Han Dynasty: '原本山川，极命草木', which was written roughly in 200 BC. It means "exploring the ultimate origin of nature like mountains, rivers, grass, and trees." The interest to explore knowledge in the physical world has been rooted in the Chinese culture.

Declaration

The work in this thesis is based on the research carried out at the Department of Information Engineering, University of Padova, Italy to fulfill the requirements for the degree of Doctor of Philosophy under the supervision of Prof. Massimo Melucci and Dr. Emanuele Di Buccio. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is completely my own work unless referenced to the contrary in the text.

Benyou Wang

Abstract

In modern Natural Language Processing (NLP) and Information Retrieval (IR), individual words are typically embedded in vector space, called 'word vectors' or 'word embedding', to enable differentiable optimization in neural networks. This leads to a new NLP paradigm that could deal with individual words in neural networks.

The first issue of the above paradigm is that components in neural networks (like word vectors and hidden states) usually do not convey any concrete physical meaning. One typical way is to use probabilities as well-constrained quantities to better understand neural network components. The challenge of traditional probability theory is that it cannot treat words as atomic discrete events since words are embedded as dense vectors that are not necessarily mutually orthogonal. This thesis proposes a novel framework based on Quantum Probability Theory (QPT) that defines probability axioms in vector space, to probabilistically ground word representation, semantic composition, and semantic abstraction in a unified space.

Another issue of the paradigm is that the inductive bias of learning word vectors relies on only the distributional hypothesis: *linguistic items with similar distributions have similar meanings*, while other aspects are usually ignored. This thesis focuses on one of the most nontrivial aspects, namely the spatially or temporally sequential aspect of words. The spatially sequential aspect refers to capture the spatial position of words in any bag-of-words document encoders, while the temporally sequential aspect refers to mine the time-specific word meaning in the scenario when word meanings may evolve with time. Interestingly, the complex-valued word embedding (with amplitude terms and phase terms), which is induced from QPT, could be naturally used to model sequence (both for spacial sequence and temporal sequence) by directly encoding sequential order in phase terms. The benefit is that the rotation nature of phases in waves makes sequential encoding being always bounded no matter how long the length of the sequence/dynamics is.

Furthermore, a side effect of the thesis is to bridge the gap between *complex-valued word embeddings* and *sinusoidal position embedding*; it therefore reinterprets commonly-used yet 'magic' sinusoidal position embedding in a principled way: sinusoidal position embedding is a real-valued variant of the proposed complex-valued word embeddings. Beyond the spatial dimension, the thesis also explores sinusoidal embeddings in temporally-sequential dimension, called 'Word2Fun', for the temporal evolution of words. Word2Fun is proved to be able to approximate any continuous word meaning evolution.

The thesis implements the QPT framework with 1) a Quantum Probability Driven neural Network (QPDN) for document modeling that achieves comparable performance with SOTA approaches in text classification benchmarks; and 2) a further extension for text matching, called 'complex-valued network for matching' (CNM) , that achieves comparable performance with SOTA approaches in question answering (a typical matching task) benchmarks. This additionally shows the potential to use complex-valued word embedding in general document representation. For the complex-valued word embedding in sequential

modeling, the empirical study also evidences the superiority of the 'complex-valued word embedding' in spatial sequence modeling and Word2Fun in temporal sequence modeling.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivations

Natural language is one of the most important interfaces between intelligent computer applications and end users or experts users such as journalists, linguists, or social scientists. The users may express information needs using natural language whereas intelligent computer applications usually display in textual format the information considered relevant to the users' information needs. These applications include but are not limited to automatic text classification [91], question answering systems [70], topic monitoring [40], [47], and recommender systems [27].

Interestingly, journalists, linguists, and social scientists may carry out longitudinal analyses involving textual corpora spanning long periods, the latter activity being investigated and supported by the computational social science community [5]. The longitudinal analysis involving textual corpora spanning long periods is challenging because the meaning of a word may change over time to an extent that the meaning of one word in a certain period of time can be very different from the meaning of the word in another period of time. Therefore, investigating the time-aware dynamic aspect could be beneficial to the approaches, the methodologies and the systems adopted to carry out longitudinal analysis.

Moreover, the users may want to transparently understand the working mechanisms of computer algorithms which are usually based on deep learning, in other words, the users may need some explanation of the results yielded by an algorithm; a method that helps to transparently understand the working mechanisms of algorithms may help one to know whether these algorithms introduce risks or biases at prediction time, for example.

This thesis proposes a methodology that aims to improve the effectiveness, interpretability, and efficiency of word representations, especially in the scenario where the meaning of words can evolve over time or can depend on the position in text.

Word representation is a cornerstone in Information Retrieval (IR) and Natural Language Processing (NLP). In IR, individual words such as query terms or document words are the basic features used by many retrieval models, e.g. the classical probabilistic model [105] or language modeling-based approaches [153]; in these models two arbitrary words in the feature space are typically assumed to be independent of each other[1] — this is called

[1]There are some variants of the vector space model and some language model-based approaches without independence assumption. Bi-gram language modeling-based approaches introduce dependence structures in the language models [39], whereas the vector space model can utilize non-orthogonal basis vectors. However, they are computationally expensive and therefore they are not largely used. The aforementioned models allow to model dependence but at the expense of computational cost; for example, the vector space model would naturally allow word dependence by using non-orthogonal basis vectors, which were costly at

"independence assumption".

Following the pioneering work of [14], representing words as dense vectors in vector space becomes a new paradigm [89], [97]. In such a paradigm, word vectors are not necessarily orthogonal to each other and it, therefore, does not follow the 'independence assumption'. This provides some flexibility to model semantic relationships between words. Recently, pre-trained language models [31], [103], which adopt 'contextualized word vectors' [98],[2] have been largely improving the State Of The Art (SOTA) in IR and NLP by extending static word vectors to context-aware word vectors.

Although word vectors have empirically been demonstrated to be effective in various NLP benchmarks, two open issues require further investigation [131]: (1) word vectors lack interpretability especially when they are used to model documents with Deep Neural Networks, e.g., Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) or Transformers; (2) word vectors themselves are limited to solely capturing co-occurrence information while other aspects, e.g., word order, time-aware semantic evolution, sentiment polarity, compositionality[3] are usually ignored.

Since Neural Networks can better deal with vectors than discrete tokens like words, words usually have to be transformed to dense vectors by means of word embedding.[4] By using a word embedding layer as the basic feature layer, researchers tend to shift, in terms of NLP paradigm, from *feature engineering* to *architecture engineering* [78] which focuses on the design of suitable network architecture. Various complicated Neural Networks architectures have been proposed, for example [22], [41], [84], [113], [117], [127]. However, a Neural Network architecture makes NLP models much more complicated and hardly interpretable.

Interpretability can be characterized by two different aspects [77]: *transparency* that aims to reveal the working mechanism of models, and *post-hoc explanation* that aims to better know the learned information from a well-trained model. The complexity of NLP models increases the difficulty of dealing with both these two aspects of interpretability. The attention mechanism, e.g., [9], which adopts an explicit probability distribution among all input tokens, was arguably claimed to be more interpretable in neural networks [57], [112], [140]. Inspired by the benefit of using the attention mechanism, the approach proposed in this thesis provides a probabilistic interpretation for semantic representation, semantic composition, and semantic abstraction in a unified framework.

Specifically, this thesis proposes to use the Quantum Mechanical Framework (QMF) to convey concrete probability-grounded meaning for each component of a Neural Network:[5] such concrete meanings for components could contribute to an aspect of *transparency*, called 'decomposability'[6] as explained in Sec. 2.2.1. Interestingly, the prevalent paradigm that represents words in vectors space do not admit a previous independence assumption

---

that time and not consistently more effective than orthogonal basis vectors.

[2]The neighboring words in the sentence/document are considered as the 'context'. In contextualized word vectors, the vector of a word depends on its concrete context and it therefore varies in different sentences/documents. Instead, previous word vectors are static in all contexts.

[3]Compositionality refer to how words semantically compose with other words.

[4]An 'embedding' is generally a mapping. This thesis denotes the mapping from words to word vectors as 'word embedding': namely, $f : \mathbb{N} \to \mathbb{R}^D$.

[5]In this thesis, components refer to neural network layers including dense layers, RNN layers, CNN layers, etc., or building block that consists of a group of layers.

[6]In [77], decomposability is defined as a aspect of transparency at the level of individual components. It expects that each part of the model (e.g., each input, parameter, and calculation) admits an intuitive explanation. Let us take convolution neural networks (CNN) for NLP tasks [63] as an example: the embedding layer (as the input), the convolutional kernels (as the weights), convolution (as the calculation) usually does not admits meaningful explanation. Models with better decomposability are expected in NLP.

that defines each document or sentence as a set of mutually independent words, but leading to a new scenario that words, which are viewed as events in probability theory, are defined as dense vector and are not necessarily orthogonal of each other. The motivation to use Quantum Probability Theory (QPT) as defined in the QMF is due to that it directly defines events in vector space (actually a complete normed vector space) while the classical probability theory is based on sets.[65] Whereas QPT integrates the logical, probabilistic, and vector space models for IR as proposed in [126], this thesis proposes to utilize QMF for the neural network paradigm for NLP tasks. For *post-hoc explanation*, the framework unifies semantic bases (called 'sememes'), words, semantic composition, and semantic abstraction in a single vector space, thus providing some potential to interpret the learned components with readable linguistic units, e.g., words. This thesis implements the framework with neural networks called Quantum Probability Driven Networks (QPDN). The results obtained from experiments performed the text classification benchmarks demonstrate that QPDN achieves comparable performance but with better interpretability. A extension of QPDN in text matching (called 'CNM') also achieves comparable performance when using benchmarks for question answering, which is another typical matching tasks.

Differently from common NLP/IR applications based word vectors defined in the real field, QPDN naturally induces complex-valued Neural Networks components, among which complex-valued word embedding is the most representative contribution. By leveraging the naturally induced definition of Neural Network components in the complex field, this thesis utilizes the phase term in complex-valued word embeddings to model sequential aspects of words. The utilization of the complex field to embed word representations as vectors allows to model words as waves and in particular to unfold a complex number in the variable phase dimension as a sine wave for the imaginary part and as a cosine wave for the real part.

Note that the Distributional Hypothesis[48] is the only inductive bias of typical word embedding [89], [97]; the distributional hypothesis is made to learn vectors of words solely based on their context. However, there are many other aspects that also affect word representation, the most representative being sequential information such as spatial positions and temporal timestamps. When considering sequential information for words, one crucial point is that it should be able to deal with arbitrarily long sequences, and the representation should be bounded to avoid the possible gradient explosion issue.[7]

Intuitively, when using complex-valued word embedding, the rotation nature of waves allows to model the arbitrary length of sequences without considering the boundedness property, since the rotations only affect the phase and the norm is always a constant value that only depends on the amplitude. Moreover, the complex-valued word embedding also brings an ideal property that the relative (especially spatial) distance between two words could be recovered by a Hermitian dot product.[8] This thesis mainly considers two scenarios where sequential aspects need to be modeled: (1) as for the spatial case, modeling word orders in SOTA neural networks, i.e., transformer; (2) as for the temporal case, diachronic word evolution.

[7] In this thesis, we expect to make word embeddings or more generally hidden states bounded, since here we involve a variable i.e. position or time as input that could be very big. We called it a 'boundedness property' in the next sections. Traditional neural networks usually adopt some activation functions like sigmoid or tanh to guarantee the boundedness property for hidden states. However, there are some unexpected saturation areas for input in those activation functions.

[8] For two complex numbers $e^{i\theta_1}$ and $e^{i\theta_2}$, their Hermitian dot product, i.e., $\langle e^{i\theta_1}, e^{i\theta_2} \rangle = e^{i\theta_1} e^{-i\theta_2} = e^{i(\theta_1 - \theta_2)}$, results in a relative distance of phrase $\theta_1 - \theta_2$ in the exponential term. This property cold be used to capture relative distance between words in a document/sentence, or the time difference between to timestamps.

Figure 1.1 provides a pictorial representation of the motivations and the contributions of this thesis and how they relate to each other. To address the interpretation issues, this thesis proposes the aforementioned quantum-probability-driven network (QPDN) [134] and its extension in text matching (called 'CNM') [74] which naturally induce complex-valued word embeddings. Also, such complex-valued word embedding can be beneficial to model sequential aspects of words, i.e., spatial and temporal dimensions by encoding sequential aspects in the phase terms in complex-valued word embeddings [137]; this thesis proved that it is highly related to sinusoidal encoding.

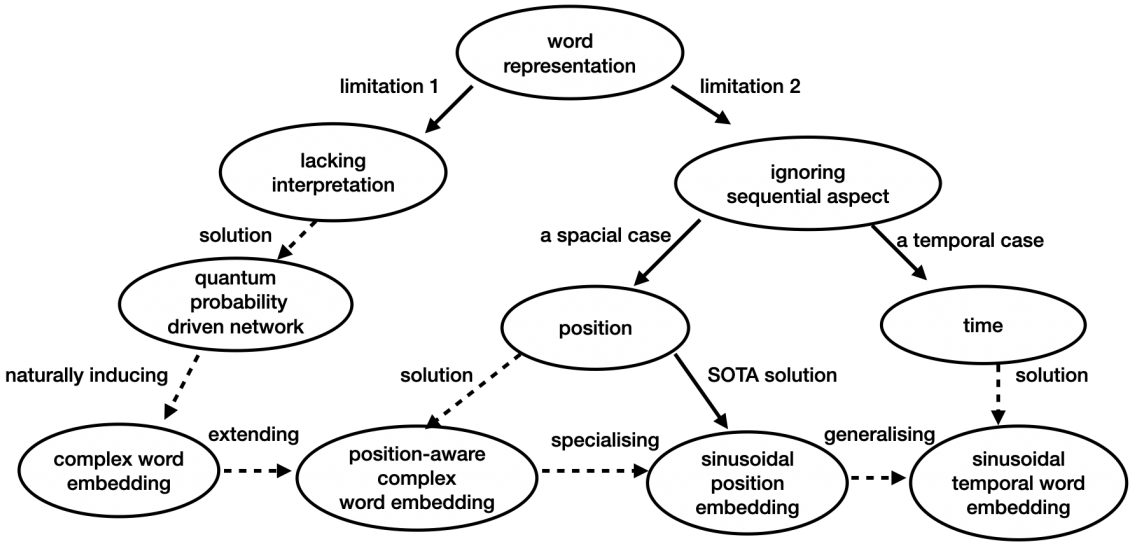


Figure 1.1: The overview of this thesis. This thesis aim to model words as particles for the Limitation 1 and encoding words as waves for the Limitation 2. Dashed lines indicate that the contributions are from this thesis.

1.2 Research Problems

In order to address the two issues introduced in Section 1.1, this thesis focuses on two research problems.

1.2.1 Modelling words as particles and probabilistic interpretation thereof

Word vectors have become the backbone for modern NLP; some downstream tasks using word vectors usually adopt CNNs and RNNs for document representation, matching, and inference. With such complicated neural networks, it is hard for end-users and system designers to understand how the models learn and predict. This may cause difficulties in interpreting typical real-life computer-based systems processing natural language such as text classification and retrieval systems.

Attention-based network architectures [9], [127] are arguably to have better interpretation than other architectures. [57], [112], [140] However, the current attention-based network architectures such as that proposed in [127] are partially probabilistic, thus limiting the interpretability power of probabilistic modeling.

Probability theory can be utilized to improve interpretability because it allows to model events by using simple mathematical structures and operators. The standard probability theory that can be utilized to this aim is Kolmogorov's as described in [65]. The probability theory defined by Kolmogorov is based on sets since every observable elementary event corresponds to a set element and every observable property also known as random variable corresponds to a subset of the set of elementary events. The Kolmogorov probability theory is characterized by the assumption that every elementary event is distinct each other, in other words, they are independent each other, the latter being a different notion from the notion of statistical or stochastic independence which is about random variables. However, the Kolmogorov theory might not fit a situation in which the words are observed because the words might not independent each other since one word can semantically be related to another word or even to a subset of words. A probability theory that gives up the so-called "independence assumption" of words is thus needed.

Quantum Probability Theory (QPT) may be of use in modeling dependent words since it is directly defined in vector spaces instead of sets. The elementary events are represented by vectors and the events are represented by vector subspaces. The vectors are related by linear function, thus yielding superposed vectors corresponding to words with different meanings at the same time. In such a way, words are treated as they were particles, the latter being the object of Quantum Physics in which the experimental outcomes are modeled by QPT. This thesis shows how the current NLP/IR paradigm may in terms of interpretability benefit from Quantum Probability Theory by treating words as particles. To this aim, this thesis defines a new semantic space with a finite set of independent sememes as elementary events, while the classical probability theory defines words as elementary events. Thus, each word could be modeled as a superposition of those elementary events that are not necessarily orthogonal. Furthermore, the semantic abstraction over these word representations could directly use projection measurement borrowed from QPT. The first research problem can be formulated as follows:

RP1: *Can a probabilistic word representation in vector space provide better interpretability and also maintain comparable effectiveness with the state of art (SOTA)?*

Since all components with probabilistic physical meaning encapsulated in a unified space, it provides some 1) transparency: each component has concrete physical meaning; 2) post-hoc explainability: each component could be explained by its connection to linguistic units like sememes and words.

1.2.2 Encoding words as waves for sequential modeling

Most existing word vector-based representations adopt a paradigm called 'one vector per word': a word vector will be statically fixed and global no matter which context it appears. No matter word embeddings are learned by neural networks [89] or decomposition [71], [97], they are learned by the inductive bias of co-occurrence. However, words themselves are not only shaped by co-occurrence. Many other aspects affect word representations. For example, in many fine-grained scenarios, words are multifaceted in other aspects e.g., sentiment polarity, word compositionality, the degree of being polysemous, domain-awareness, ambiguity, etc. This thesis considers a more non-trivial case: the aspect itself is sequential, for example, time or positions. The second research problem this thesis is addressing in this doctoral research work can be formulated as follows:

RP 2: *How to model sequential aspects of words in vector space?*

This thesis will focus on two sequential aspects: position and time. The former is introduced in Section 1.2.2.1 while the later is introduced in Section 1.2.2.2.

1.2.2.1 Spatial sequence case: Position Encoding

Word order is crucial for natural language. The earlier neural network approaches for NLP are Recurrent Neural Networks, which include the popular variant Long and Short-Term Machine (LSTM), and Recursive Neural Networks; they process natural language as a sequential or tree-like structure in which order is inherently modeled. However, both Recurrent and Recursive Neural Networks are inefficient due to the fact that sequential and tree-like structures are unfriendly to parallel. Table 1.1 reports the complexity to process a n-length sequence using various neural networks in terms of the count of serial processing steps.

models	processing steps
Recurrent NN [53]	$\mathcal{O}(n)$
Recursive NN [117]	$\mathcal{O}(\log n)$
Transformer [127]	$\mathcal{O}(1)$

Table 1.1: Parallelization efficiency. The complexity is measured by the serial processing steps to process a n-length sequence.

In order to address this issue, the community investigated how to replace recurrent and recursive structures with more parallel-friendly ones: Convolutional Neural Networks (CNNs) [41] and, recently, Transformer [127]. The drawback of the both is that, in their original formulation, they cannot properly perceive word order at the architecture level since they respectively process convolution and attentions in parallel. Therefore, CNNs and Transformers were equipped with position embeddings to perceive word order at the feature level. This thesis will mainly focus on Transformer, even if the proposed approach is also applicable to CNNs. Transformer [127] has been used in various tasks and achieved many SOTA results in NLP such as natural language understanding [31], [130], question answering [31], [104], and machine translations [127].

The Transformer is a network architecture "based solely on attention mechanisms" [127], where an attention function can be described "as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key." [127]. If a set of queries is represented by a matrix Q and keys and values respectively with the matrix K and the matrix V, the weights depend on $A = QK^T$. In [127] the authors proposed a "Scaled Dot-Product Attention":

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T/\sqrt{d_k})V \tag{1.1}$$

Such a weighted sum does not consider the word order. Thus, the word representation at position x is additionally injected with Absolute Position Embeddings (APEs) [41] and Relative Position Embeddings (RPEs) [114].

This assumes that word representations also depend on their absolute position appearing in a document. How to unify word information and position information in a single framework is an open research question. This doctoral research work investigates the use

of complex word embedding to address this question, since complex word embedding could model sequential features in the phase with a bounded norm — i.e. related the amplitudes only. A detailed discussion on these aspects will be reported in Chapter 4.

1.2.2.2 Temporal sequence case: Dynamic Word Embedding

The phenomenon that word changes with time have been investigated for many decades. An example is reported in Figure 1.2 that depicts the meaning change of the word 'president'. Observe that 'George' and 'Bush' are close to the 'president(1989)' and 'president(2001)', since the 'George Bush' as the father became the president of the US in 1989 and his son also became a president. The change of presidency is an example of word meaning change which is related to historical events. Other examples of word meaning change could be attributed to cultural shit and language usage shift [121].

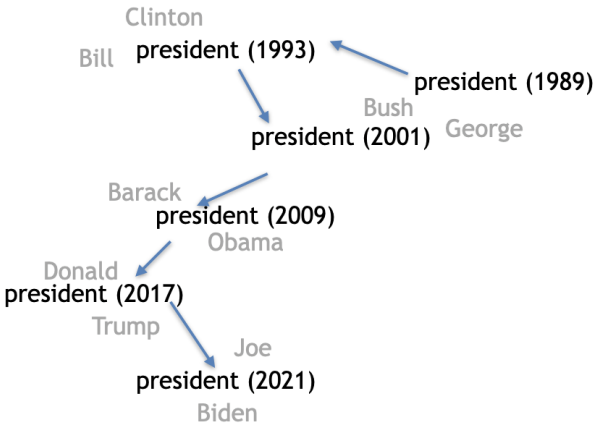


Figure 1.2: The expected meaning change of 'president' from 1989 to the present.

This thesis carried out some data exploration to investigate if word representations based on co-occurrence vary through time. Let us define a term-document co-occurrence matrix, $M \in \mathbb{N}^{|V|\times|D|}$, in which the j-th element in the i-th row denotes the number that the word w_i appearing in j-th document, namely, $M_{i,j} = \text{count}_{d_j}(w_i)$. Then one get a term-term co-occurrence matrix by $E = MM^T$. Note that there are many other ways to get a term-term co-occurrence matrix, for example, the positive point-wise mutual information (PPMI) matrix.

Let us denote the corpora in a specific time t as $\mathcal{C}_t$ and its term-term co-occurrence matrix E_t. The closeness between two time-aware term-term co-occurrence matrices can be defined as the Frobenius norm of the difference between them:

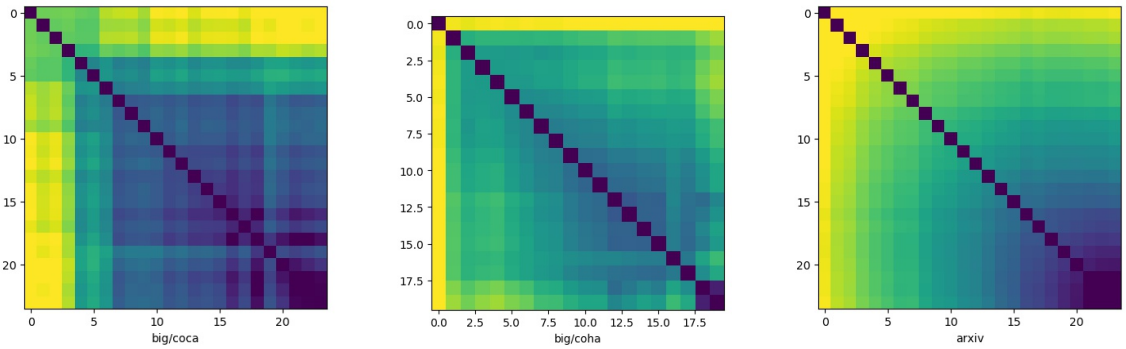
$$< E_{t_1}, E_{t_2} >= |E_{t_1} - E_{t_2}|_F$$

$< E_{t_1}, E_{t_2} >$ measures how much language usages in t_1 is similar to the counterpart in t_2. As shown in Figure 1.3, the term-term co-occurrence matrix gradually changes with time.

Diverse existing works tried to connect word representations within only two times-tamped corpora (or two bins) using extra orthogonal transformations [47], a latent diffusion process [11] or a direct regularizer [148]. A limitation of these approaches is that a transformation between $E_{(t)}, E_{(t+1)}$ is totally independent to the one between $E_{(t+1)}, E_{(t+2)}$. However, semantic change may be a gradual process and the evolution across long periods

dataset	time range
COCA	1990 - 2019
COHA	1810 - 2009
Arxiv abstract	2007.4 - 2020.4

Table 1.2: Time-stamped Dataset



(a) Yearly in COCA. 0 refers to 1990

(b) Decennially in COHA. 0 refers to 1810s

(c) Monthly in ArXiv abstract

Figure 1.3: The figures shows how term-term co-occurrence matrices differ in different time. The darker, more similar the two term-term co-occurrence matrices are. The co-occurrence matrices are calculated using the 10000 most frequent words.

should be considered as a unified process. [106] proposed to treat time as a separate variable and model words as functions over time. This could, therefore, unify all time series in a single dynamic process. Also this thesis explores modeling words as functions, specifically using sinusoidal functions, since trigonometric polynomials could universally approximate any lexical-semantic evolution over time. The approach proposed in this doctoral research work has several advantages over [106]: theoretical aspect, interpretability, empirical effectiveness, and insensitivity to the rotation-invariance issues; they will be discussed in detail in Section 4.2.5.

1.3 Overview of the Contributions

The main contributions of this doctoral research work are:

- a novel framework that is fully driven by quantum probability theory in NLP [74], [137];
- a novel paradigm to model word as functions for both spatial and temporal dimensions, which could be further extended as a general approach to encode sequence in vector space [132], [133], [137];
- a principled way to better interpret existing widely-used position embeddings [135], [137];
- the first work to bridge complex-valued embedding and sinusoidal embeddings;

- the first work in NLP to link imaginary numbers in complex-valued representations to concrete physical meanings (i.e., word order/time).

1.4 Thesis Overview

The content is organized as below.

Chapter 1 introduces the backgrounds, motivations, and contributions in this thesis. Two main research problems are proposed and will be further discussed in the latter of this thesis.

Chapter 2 summarizes the related work about word representation, from the very beginning of IR literature and recent contextualized word embedding. The shortage of existing word representation is mentioned and inspired this thesis.

Chapter 3 propose the detailed methods for the above research problem 1.

Chapter 4 propose the detailed methods for the above research problem 2.

Chapter 5 shows the experimental results.

Chapter 6 concludes this thesis and discusses the future work.

Chapter 2

Background and Motivations

Words have been considered as basic features for document representation in Information Retrieval (IR) [105], [153] and Natural Language Processing (NLP). The representation used in most approaches relies on the 'independence assumption' in which words are mutually independent. This approach has two advantages: (1) it is natural and interpretable; (2) it can benefit from efficient data structures such as inverted indexes since sparse features for words can be used as descriptors in inverted indexes and documents containing given words can be efficiently retrieved.

Following the pioneering work of [14], much research has been devoted to representing words as dense vectors [89], [97]. Moreover, the development and the availability of computing resources, especially the use of GPUs, has allowed the use of huge amounts of data to train efficiently not only dense words vectors but also neural networks that are on the top of these word vectors. This paradigm follows the "Distributional Hypothesis" [37], [48]. In [48] Harris argued that "it is possible to define a linguistic structure solely in terms of the "distributions" (patterns of co-occurrences) of its elements", and that was called 'distributional structure'. Harris's work is considered as the origination of distributional hypothesis which states that words appearing in similar context tend to be embedded closely in vector spaces; they could be first trained by self-supervised learning on large-scale plain corpora and then be transferred to a special (typically small-scale) downstream domain. In this paradigm, word vectors are not necessarily orthogonal to each other and they don't follow the 'independence assumption'.

In NLP, word vectors [14], [89], [97] has been widely used in lexical tasks like word analogy and synonym detection. By transforming word tokens to word vectors, previous works [63], [67] build neural networks on the top of word vectors since word vectors are differentiable and word tokens are not. These neural network approaches largely boost the performance for NLP tasks including text classification [91], natural language inference [22], question answering [151]. More recently, a 'contextualized' word embedding [31], [98] has been used in various tasks [104], [130] and achieved the SOTA results .

In typical IR scenarios, most dense vector based methods were initially adopted in the re-ranking phase; few of them worked on the pre-ranking phase that involves inverted index since the inverted index cannot provide efficient access to these dense representations. Recently, word vector based neural networks can be also be used in the pre-ranking phase thanks to vector search approaches, i.e., Maximum Inner Product Search (MIPS) [115]. Moreover, recent works [61], [76] use dense vector representation to substitute inverted indexing [58] in pre-ranking (recall) phase, in order to make use of word vectors based document representation and ranking.

Although word vectors are successfully applied to many applications in IR and NLP.

Compared to BM25 [105] or Language Model [153] in IR, word vectors are less interpretable. To this end, we therefore propose a new framework based on quantum probability to reinterpret many concepts of natural language in a unified vector space.

Generally, word vectors themselves are limited to solely capturing co-occurrence information. However, in many fine-grained scenarios, words are multifaceted and many other aspects, e.g., word order, time-aware semantic evolution, sentiment polarity, compositionality, etc., are usually ignored. Interestingly, some of these aspects are sequential like word positions and word orders. How to encode these sequential aspects in vector space is non-trivial and it is worth investigating.

In this chapter, we will introduce some representative methods to represent words as vectors and discuss their limitation in terms of interpretability and multifaceted modeling.

2.1 Representing Words in Vector Space

2.1.1 Word representation in early IR and NLP

In early IR and NLP research, each word was customarily assumed to be an independent element in language universal, which was called the 'independence assumption'. For example, we represent a document d with a word sequence $w_1, w_2, \cdots, w_L$, as its TD-IDF vector $\vec{d} \in \mathcal{R}^V$,[1]

$$\vec{d}_{w_i} = \begin{cases} \text{TF}_d(w_i) \times \text{IDF}(w_i) & \text{if } w_i \in d \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

V is the number of all words. $\text{TF}_d(w_i)$ is the frequency of the word w_i in document d and measures how much the word appears in the document; if normalized over the document length, $\text{TF}_d(w_i) = \frac{\#(d,w_i)}{\sum_{w_j \in d} \#(d,w_j)}$. $\text{IDF}(w_i)$ is the Inverse Document Frequency [119] of word w_i and provides a measure of the word "specificity", i.e., how discriminant the word is in the collection: $\text{IDF}(W_i) = \log \frac{|D|}{\#_{d_j \in D}(w_i \in d_j)}$. More advanced retrieval methods have been proposed, e.g., BM25 [105] or Language Modeling [153].

Vector representation of words based on TF-IDF or BM25 weights is sometimes categorised as a 'one-hot vector' or 'descriptor', in which a word w_i is represented as a V-length[2] vector with a single value equal to one (the i-th element corresponding to the w_i) and zero elsewhere. Thus, the TF-IDF or BM25 vector representation of a document is the linear combination of the one-hot vectors representing the occurring words, where the coefficients are the TF-IDF or BM25 weights. The benefit of the above methods based on 'independence assumption' is that the results are interpretable. For example, once a document is retrieved, one can easily tell the end user how much each word contributes to the relevance prediction.

Although the above methods are effective in ad-hoc retrieval scenarios, they are less effective in more complicated IR tasks that do not solely rely on exact lexical matching. Table 2.1 from [72] shows some simple examples about the gap between semantic matching and exact matching; this gap motivated the community to develop some word representation approaches that do not strictly follow the 'independence assumption'.

Typically, document candidates may not exactly match the given query, and that leads to a 'partial term matching'.

[1] Here, we do not consider the smooth.

[2] V is the size of the whole selected word vocabulary. Sometimes one may prune some stopwords and low-frequency words.

For instance, there is a partial term matching between the query 'china kong' and the document 'china hong kong'. However, they are totally different named entities: the former refers to an America actor,[3] while the latter refers to a city in China.[4]

In Table 2.1, the query and document pairs in the top three cases of partial term matching are correctly semantically matched, while the two bottom cases are incorrectly matched. The 'partial term matching' issue becomes challenging under the independence assumption.

[72] claims that another category of word representation, namely, word embedding (a.k.a. word vectors) based on the 'distributed representation', may be beneficial for the "partial term matching" issue. Approaches belonging to this category of word representation will be introduced in the following subsection.

query	document	term matching	semantic matching
seattle best hotel	seattle best hotels	partial	yes
pool schedule	swimming pool schedule	partial	yes
natural logarithm transformation	logarithm transformation	partial	yes
china kong	china hong kong	partial	no
why are windows so expensive	why are macs are so expensive	partial	no

Table 2.1: Difference between exact (term) match and semantic match [72]

2.1.2 The Distributional hypothesis for word representation

Word embedding is driven by the *Distributional Hypothesis* [48]. The core of distributional hypothesis states that linguistic items with similar distributions have similar meanings and hence words with similar distributions should have similar representations. The distributional property is usually induced from document or textual neighborhoods (like sliding windows). While the *Distributional Hypothesis* was proposed many decades ago, the techniques of word embedding trained in a neural network has a much shorter history of about one and half decades [14].

Suppose there exists V words in total and each word is indexed with a specific integer $i \in \mathbb{N}$, more specifically $i \in [1, V]$. Word embedding aims at learning a one-to-one mapping from each word to a D-dimensional vector as below:

$$f : \mathbb{N} \to \mathbb{R}^D \tag{2.2}$$

Differently from the representation discussed in Section 2.1.1, where each word is encapsulated in a specific dimension of vector space (a.k.a., *one-hot representation*) and individual words are assumed to be independent, in the word representation in Eq. 2.2 each word is embedded in all dimensions of vector space using a distributed manner (a.k.a., *distributed representation*) that can model possible relationships between words.

There are typically two ways to obtain word vectors [12], [71]: one way is that adopted by *count* models, among which Convolution Composition of the Memory Vector [92], Latent Semantic Analysis (LSA) [54], [68], and Hyperspace Analogue to Language (HAL) [82] are representative early works. Another way is prediction-based neural methods, which have become more popular recently [12]. Some popular word embeddings relying on the Distributional Hypothesis are reported below:

[3] See `https://en.wikipedia.org/wiki/China_Kong`

[4] `https://en.wikipedia.org/wiki/Hong_Kong`

model	type	architecture	task
NNLM [14]	static	single-layer MLP	$(a, b) \rightarrow c$ predicting the next word
Skip-Gram [89]	static	single-layer MLP	$b \rightarrow c, \quad b \rightarrow a$ predicting neighboring words
CBow [89]	static	single-layer MLP	$(a, c) \rightarrow b$ predicting central words
Glove [97]	static	single-layer MLP	$\vec{w_a}\vec{w_b}^T \propto log(p(\#(w_a w_b)))$ predicting the log co-occurrence count
ELMO [98]	contextualized	LSTM	$(a, b, c, d) \rightarrow e, \quad (e, d, c, b) \rightarrow a$ bi-directional language model
BERT [31]	contextualized	Transformers	$(a, \texttt{[mask]}, c, \texttt{[mask]}, e) \rightarrow (\_, b, \_, d, \_)$ predicting masked words
GPT [103]	contextualized	Transformers	$(a, b, c, d) \rightarrow e$ predicting the next word

Table 2.2: Typical ways for word vectors and language models. (a, b, c, d, e) is an example text sequence. ELMO, BERT, and GPT usually work on much longer sequence than NNLM, Skip-gram and CBOW.

- **NNLM** (Neural Network Language Model) [14] preliminary aims to build a language model, while learning word embedding is not the main target. However, this is the first work in learning word vectors in a neural network.
- **Skip-Gram** [89] balances a trade off between performance and simplicity. Skip-Gram uses a word to predict one of its neighboring words.
- **Cbow** [89] uses context words to predict the current word. The difference between Skip-gram and Cbow is that in order to predict the target word, Cbow uses many words as the context while Skip-Gram uses only one neighboring word.
- **Glove** [97][5] takes advantage of global matrix factorization and local context window methods. It is worth mentioning that [71] explains that the Skip-gram with negative sampling derives the same optimal solution as matrix (Point-wise Mutual Information, PMI) factorization.
- **ELMO** [98] trained a multiple-layer LSTM on a bi-directional language model which not only predicts the next word but also predicts the previous word. This work proposes a two-stage paradigm: first pre-training in self-supervised tasks and then fine-tuning on downstream supervised tasks. Note self-supervision makes it feasible to train on the vast amount of plain corpora without annotations. This results in 'contextualize word embeddings', since, in downstream tasks, embedding for words is not static but generated in real-time way depending on the context.
- **BERT** [31] is a deep bidirectional language model trained on masked language model and next sentence prediction. This work also adopts the two-stage pre-training and fine-tuning paradigm. [31] has largely improved the SOTA of many natural language understanding tasks including text classifications, sequence labeling, and question answering.

[5] https://nlp.stanford.edu/projects/glove/

- **GPT** [103] is a language model trained. The difference between BERT and GPT is that the former has an encoder architecture while the latter has a decoder architecture. This allows the latter to perform on generation tasks including summarization and generation-based question answering.

The overview is shown in Table 2.2. Many typical methods can be used to build word vectors from plain corpora. For example, [89], [97] build a one-to-one mapping between words and their vectors, which is called 'static word embedding' since it is static and not related to word context. Skip-Gram, CBow, and Glove [89], [97] adopt linear architecture to conduct calculations between word vectors, resulting in efficient training. Inspired by [98],[6] many pre-trained language models adopt 'contextualized word embedding' to model words in a specific context. Pre-trained language model [31] is a new two-stage paradigm for natural language tasks to usually train with a self-supervised meta-task in task-agnostic corpora (e.g., masked language model and casual language model), and then fine-tune a (usually small-scaled) specific downstream tasks. As shown in Table 2.2, recently, BERT [31] and GPT [103] propose to use multiple layers Transformers as basic architecture, but that led to much more parameters.

For 'contextualized word embedding', the vector for a word depends on its specific usage in a context. For example, the meanings of '`bank`' in '`river bank`' and in '`money bank`' are supposed to have some difference; the 'contextualized word embedding' may arguable help word meaning disambiguation. In BERT and GPT, subwords (limited set of common sub-word units, e.g., 'wordpieces' [143][7]) are used to reduce the size of word vocabulary and therefore save parameters; note that this also beneficial to handle rare words.

2.2 Limitations of Word Vectors

2.2.1 Interpretability

Interpretability is a crucial issue in modern Machine Learning, especially to achieve trusty, causal,[8] and transferable[9] AI. In [77] Lipton defines interpretability being twofold: transparency and post-hoc explanation. The former answers the question *how does a model work?*, while the latter relates to the question *what can we know from a trained model?*. Section 2.3 will discuss the difference between these two aspects of interpretability.

In NLP scenarios, some downstream tasks using word vectors usually adopt CNN and RNN for document representation, matching, and inference. With such deep neural networks, it is hard for either end-users or system designers to understand how the models learn and predict. This may cause many issues in typical real-life systems.

Generally, interpretability could benefit two groups of people: system designers and end-users. For example, given some "bad" cases, system designers could post-edit trained models to avoid fatal consequences introduced by such cases, if they know the models in a transparent manner. Interpretability might help system designers to transfer trained models to other domains. Recently, some concerns have been raised about models' bias

[6] A earlier work from Melucci [85] also suggests to utilize distinct basis vectors for each word depending on context.

[7] For example, a word `acknowledgment` is divided into two subwords `acknowledge` and `##ment`, in which `##` means that the current subword is not an independent subword.

[8] Typically, a machine learning model aim to mine the association between two factors, but the two factors are necessarily a cause and an effect.

[9] Transferability denotes the ability to generalize learned models to other domains.

	transparency	post-hoc explanation
question	*how does a model work?*	w*hat can we know from a trained model*
time	before or during training	after training
checkpoint-agnostic	yes	no

Table 2.3: The difference between transparency and post-hoc explanation. *Checkpoint-agnostic* indicates whether a property should be examined with respect to a specific checkpoint.

and stereotypes which could amplify existing societal bias and stereotypes [156], [157]; models with worse interpretation may suffer more from this effect. As for end-users, they may ask models for prediction reasons; for example, end-users might be willing to know if those models used features that involve personal privacy.

Unfortunately, widely accepted quantitative metrics for both aspects of interpretability do not currently exist. Transparency in [77] is considered at three levels: *simulatability* at the level of the entire model, *decomposability* at the level of individual components (e.g. parameters), and *algorithmic transparency* at the level of the training algorithm. More specifically,

1) *Simulatability* refers to 'a human could take the input data together with the parameters of the model and in reasonable time step through every calculation required to produce a prediction' – this also means that model parameters are usually small-scaled for transparency; [77] claims most models including linear models, rules, decision trees, and deep neural networks are not interpretable in this sense.

2) *Decomposability* refers to that 'each part of the model - each input, parameter, and calculation - admits an intuitive explanation'; this admits that inputs themselves should be individually interpretable – this is not the case for most embedding based models since embedding itself is unknown and totally driven by data instead of symbolic common sense.

3) Models with *algorithmic transparency* should arguably have known error surface and unique converged solution if it has; this is difficult for most deep neural networks.

Informally, transparency of a model is that *we can mentally simulate (at the level of the entire model) and intuit the meaning of its components (at the level of individual components), and meanwhile know its error surface and unique converged solution if it has (at the level of the training algorithm).* Since *simulatability* and *algorithmic transparency* is arguably impossible for most modern machine learning models especially neural network models, this doctoral research work mainly focuses on the *decomposability* of individual components for better transparency: we want to convey concrete physical meaning for each component/parameter. To this end, we redefine words, semantic compositions, and semantic abstract as quantum states, mixture, and measurements thanks to QPT.

Note that simple models (like linear classifier models) do not necessarily lead to better decomposability in NLP. Linear models may heavily rely on hand-engineered features to achieve performance comparable to neural networks (such as RNNs). However, complicated hand-engineered features may negatively affect models' decomposability. In this thesis, the proposed method is implemented by a neural network which tends to operate on raw or lightly processed features. Therefore, features of the proposed method are defined in

intuitively meaningful space, which also benefits post-hoc reasoning in terms of visualization or verbalization (or called 'text explanation') [77].

In [77] Lipton gave also three examples for post-hoc explanations: *text explanations*, *visualization*, and *explanations by example*. One challenge of post-hoc explanations in Neural Network is that semantic abstraction is complicated. In this thesis, the process of 'converting raw features to high-level features' is called 'abstraction'; for example, CNN, RNN or Transformer layers which work on the top of word features/embeddings are the 'abstraction' components. Typically, such 'abstraction' components are not interpretable. For instance, we generally cannot know the meaning of a CNN kernel or an RNN cell. In this work by unifying many components like words and semantic abstraction – measurement subspace in our framework – in a single vector space, we could provide *text (word) explanations* for semantic abstraction components. The latter two post-hoc explanations – visualization and explanations by example – are trivial since they could be available for most models.

2.2.2 Multifaceted aspects of words

2.2.2.1 Various inductive biases for word embedding

The inductive bias of learning word embedding [89], [97] relies on the distributional hypothesis [48] that is intuitively stated by [37] as below:

> *A word is characterized by the company it keeps.*

By following the distributional hypothesis, the objective for word embedding is either to predict neighboring words in a local window, or factorize a co-occurrence matrix which is counted by considering co-occurrence context; [71] claims that word vectors are generally equivalent to factorize a word-context co-occurrence matrix in an implicit way. The above inductive bias is reasonable in general since some coarse-grained NLP scenarios, e.g., some classification tasks (like domain/topic classifications), can be effectively addressed by exploiting co-occurrence based semantics; some bag-of-words models (e.g.. [7], [60]) could achieve competitive results compared to SOTA models that are order-sensitive. However, words themselves are not only represented by co-occurrence. We show some basic aspects of word vectors other than co-occurrence.

In many fine-grained scenarios, words are multifaceted and the expected inductive biases is not limited to the distributional hypothesis. Many other aspects may need to be considered in word embedding, e.g., word order, time-aware semantic evolution, sentiment polarity, word compositionality, domain-awareness, and ambiguity (the degree of being polysemous).

Sentiment polarity One famous example is that word vectors cannot distinguish antonym words, e.g., 'good' and 'bad'. Based on co-occurrence context, they may appear in a similar local context, leading to similar word vectors. This may harm the performance of downstream sentiment classifications with antonym-unaware word vectors.

Word compositionality Word compositionality refers to how words compose into a phrase. Most words semantically compose in a linear way: the meaning of a phrase is a linear combination of words inside. For example, a 'hot pizza' is a pizza that is newly prepared and its temperature is still high. However, semantic composition for phrases does not necessarily behave like this. For example, a 'hot dog' is typically not 'a dog that feels hot in summer or near a heater'; instead, it is 'a kind of food consisting of a grilled or

steamed sausage served in the slit of a partially sliced bun', of course, it is not 'cooked dog'. When modeling words in vector space, we may also consider compositionality.

Ambiguity and domain-awareness Ambiguity refers to the degree of a word being ambiguous/polysemous. Let us recall the example of 'bank' in Sec. 2.1.2, one may measure the distance between 'bank' and 'money'. The difficulties come from the fact that 'bank' can be 'a land alongside or sloping down to a river or lake', or 'a long, high mass or mound of a particular substance e.g., money'; its meaning may depend on domain or context. Without knowing the domain or real context where the word appears, one cannot precisely represent its meaning.

Sequential aspects of words By representing words in vector space, it is nontrivial to capture their sequential aspects, for example, spatial positions of words or temporal timestamp of words. *The former* matters since the authors of [59] find that both word context and word order information are beneficial to build word embedding. There exist many works to encode word order during the process of training word embedding, such as the circular convolution-based memory model called 'BEAGLE' [59], random permutation [109], Embeddings Augmented by Random Permutations (EARP) [25]. However, order-aware components for training word embeddings are usually discarded in downstream tasks due to the simplification or parallel purposes; instead, other neural components like convolutions in Convolutional Seq2seq [41] and self-attentions in Transformer [127] are usually adopted. In such a scenario, only word embeddings are used without order-aware components; therefore we also need to capture word order. Especially, Transformer [127] is a bag-of-words model at the architecture level and capturing word order at the feature level is essential [137]. *The latter* – temporal timestamps of words – matters when considering the phenomenon word meaning could change over time, also called 'language change' [2]. Since language could be treated as a dynamic system that constantly evolves and adapts to the needs of its users and their environment [2]. See a concrete example in Fig. 1.2.

2.2.2.2 Motivations to capture sequential aspect of words

Note that some of the above aspects – i.e., sentiment polarity, word compositionality, and ambiguity – can be captured to some extent by contextualized word embeddings [31], [98], since making use of longer context could disambiguate these aspects. For example, the long context may provide some hints for words' sentiment polarity. Modeling sequential aspects is nontrivial and cannot be captured with additional context (i.e., surrounding words for a given word) using contextualized word embeddings [31], [98].

We will focus on two cases for sequential aspects of words, namely, *word order* and *time*. The former is crucial for word representation in order-insensitive models, e.g., CNN and Transformer; such order-insensitive models that are friendly to parallel – this is the reason why order-insensitive Transformer is widely used everywhere. The latter, i.e., time, could help linguists know how language evolves over time. One could also use such time-aware word representations to investigate cultural evolution and to perform dynamic content monitoring, which is beneficial to support specialists (end-users) such as journalists or social scientists, e.g., studying how relevant societal issues are discussed by the public over time.

More generally, encoding sequential aspects of the embedded object in vector space could accelerate training by using parallel processing. This could be used for general (especially large-scale and efficient) time series prediction. However, encoding order into

vector representation is a nontrivial problem; in this thesis, we will show how a wave-based representation of words allows sequential information to be encoded.

2.3 Modeling Words as Particles for Better Interpretation

[77] claims that one may expect to understand the meaning for components in neural networks such as input, weights, and calculation – also called decomposability in [77]. In NLP, these components include but are not limited to word embedding as input, weights in LSTM gates or kernels in CNN as network weights, and cell updating or convolution as calculation. Tab. 2.4 shows these components in RNN [23] and CNN [63], [67] for NLP. We argue that it is hard for humans to associate a concrete meaning to these components in RNN and CNN. For example, if we consider weights in LSTM gates or a convolution kernel, it is impossible to know how they are related to the semantic space.

Components	Functions	RNN (e.g., LSTM)	CNN
input	lexical representation	word embedding	word embedding
weights	parameters	weights in gates	convolution kernels
calculations	semantic composition/abstraction	cell update	convolution
output	hidden states	cells	pooled feature maps

Table 2.4: Components of Neural Networks for NLP.

Probabilities are relatively easily understandable compared to unconstrained hidden states in neural networks. Recently, the commonly-used attention mechanism [9], which adopts an explicit probability distribution among all input tokens, was arguably claimed to be more interpretable in neural networks [57], [112], [140].

The approach proposed in this thesis for word representation aims at providing a probabilistic interpretation for semantic representation, semantic composition, and semantic abstraction in a unified framework.

When word representation is based on the 'independence assumption', each word (or n-gram) can be considered as an independent and discrete event in a sample space. Classical probability theory which is based on set theory can calculate probability measures on the defined events, for example counting its frequency. However, it does not work when words are embedded in dense vectors: words can no longer be considered as mutually exclusive events.

Moreover, it also is limited that probability measurements cannot be based on infinite states. When considering the 'independence assumption', there exist countable elementary events since there are definite words (or word combinations), namely, $\{w_i\}_{i=1}^{V}$. However, under the distributional hypothesis, there exist V D-dimensional vectors for V words:

$$\mathcal{V} = \left\{ \vec{w}_i \right\}_{i=1}^{V}, \ \vec{w}_i \in \mathbb{R}^D \tag{2.3}$$

One may also expect to conduct probability measurements on a 'dummy' vector that corresponds to a 'fake' word or a specific semantic vector.[10] Let us say, any linear

[10] One example driven from the additive property of word embedding [89] is like, $\vec{women} - \vec{men}$ could be a 'dummy' vector that corresponds to 'femalizing something', which is closed to $\vec{queen} - \vec{king}$

combination of $\mathcal{V}$ may be such a dummy word vector; the possible states in a semantic space could form a set including infinite elements [11] such as

$$\mathcal{V}' = \left\{ \sum_{i=1}^{i=V} \theta_i \vec{w}_i \middle| \theta_i \in \mathbb{R} \right\} \tag{2.4}$$

In downstream tasks, one may want to measure an event with respect to a 'fake' word in $\mathcal{V}'$. One can conclude that $\mathcal{V}$ is proper subset (with V finite elements) of $\mathcal{V}'$, the former is a specific case when coefficient vector $\{\theta_i\}_{i=1}^{V}$ of the latter becomes a 'one-hot' vector [33], namely, one of V elements equals to one and zero elsewhere. The infinite nature of state space many bring difficulties for existing classical probability theory. A more general probability theory defined in vector space is needed.

The basic rationale of the proposed approach is to obtain word representation based on low-level components, namely, sememes. Sememes are the minimal atomic linguistic units that cannot be decomposed into smaller semantic units. A word is considered as a combination of one or many sememes. The quantum mechanics framework allows this goal to be achieved through the concept of "superposition"; the sememe vectors are orthogonal but not the word vectors. By doing so, one could define infinite existing or 'dummy' words that correspond to a specific coefficient vector for sememe combination.

More formally, by shifting from classical probability theory to quantum probability theory, one could transform a set of words to a semantic vector space contains existing or dummy words. The former is a set while the latter is a vector space. The fundamental difference between them was stated by [33]: a set is a primitive collection in which its elements cannot be combined together [46], whereas a space is a set in which the points (i.e., the vectors) can be mathematically combined to obtain other points of the same space [45], e.g., vectors of dummy words. The use of the sememes is also connected to [32], [33] since the sememes can be adopted as the basic features to represent terms.

Sec. 2.3.1 introduces quantum probability theory. Sec. 2.3.2 introduces how this thesis model natural language like particle within the QPT framework. Sec. 2.3.3 introduces the related work and the difference with this work.

2.3.1 QPT: a probability theory in vector space

While quantum probability theory is directly defined in vector space and could directly measure an event in vector space and events are represented as subspace that are not necessarily mutually exclusive thanks to the superposition principle. We argue that quantum probability theory better fits neural networks with word embedding, in which words are represented as dense vectors that are not necessarily independent/orthogonal, and can still be measured by subspace projections.

2.3.1.1 Quantum Physics

Quantum Physics is a fundamental theory to describe physical properties of nature at the micro scale (e.g., atoms, photons, and electrons). One basic principle in Quantum Physics is that micro particles are indeterminate before measurement; that is to say, particles are in so-called 'superposed states': particles may not exactly be in one of the basic states and may exist in many states at the same time with respect a probability distribution that can be inferred by many measurements.

[11] Each fake word has a specific coefficients $\{\theta_i\}_{i=1}^{V}$ that could be arbitrarily real numbers range from 0 to to 1 and their sums equals to one; one could know the elements of $\mathcal{V}'$ is infinite.

Considering an electron as an example, an electron has two possible configurations (i.e., up and down), which are the two mutually exclusive elementary outcomes that are considered as two *basis states*, denoted as $|0\rangle$ and $|1\rangle$, respectively [12]. Before the observation (measurement), the configuration of an electron is indeterminate.

Superposition Any *pure* quantum state can be represented as a sum of distinct basic states. For instance, a qubit has two distinct basis states ($|0\rangle$ and $|1\rangle$). In this two-dimensional case, a state of a qubit $|\phi\rangle$ is a linear combination of basis states $|0\rangle$ and $|1\rangle$:

$$|\phi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle , \tag{2.5}$$

where α_0 and α_1 are complex values, $0 \leq |\alpha_0|^2 \leq 1$, $0 \leq |\alpha_1|^2 \leq 1$ and $|\alpha_0|^2 + |\alpha_1|^2 = 1$. If $\alpha_0 = 0$ or $\alpha_1 = 0$, $|\phi\rangle$ falls onto either of the basis states with probabilities α_0 and α_1 respectively – this called the 'born rule'. Otherwise, we call $|\phi\rangle$ a *superposition* of the states $|0\rangle$ and $|1\rangle$ or equivalently a *superposition state*. The scalars α_0 and α_1 denote the probability amplitudes of the superposition.

In general, the basis states form an orthogonal basis of a multi-dimensional complex vector space, i.e. *complex Hilbert Space*. Since the coefficients are complex scalars, a superposition state is also a unit complex vector on the complex Hilbert Space.

Mixture A physical system with many particles is called a *mixture* of particles. The probability distribution of the system is represented by a *mixed state*, admitting a representation as a *density matrix*, which is a square positive semi-definite matrix with unitary trace. Technically, a density matrix is positive semi-definite with unitary trace, defined as

$$\rho = \frac{1}{n} \sum_{i}^{n} |\phi_i\rangle \langle\phi_i| \tag{2.6}$$

ρ is therefore a mixture of pure states $\{|\phi_i\rangle\}_{i=1}^{n}$. Since the states are represented as complex vectors, ρ is a complex-valued density matrix on the same space.

2.3.1.2 Quantum probability

To probabilistically describe such physical systems at micro level, quantum probability provides a sound explanation for the phenomena and concepts of quantum mechanics, by formulating events as subspaces in a vector space with projective geometry. The difference between quantum probability and classical probability is rooted in the measurable spaces on which the probability measure is defined. In particular, the classical probability is defined on subsets of a set of elements subject to union and intersection operations. Likewise, the quantum probability is defined on subspaces of a Hilbert Space with properly defined meet and join operations.

Events In quantum probability, *events* are defined as projectors on subspaces. A projector is mathematically a square matrix (a.k.a., density matrix or density operator) P such that $P^2 = P$. In particular, an *elementary element* is a 1-dimensional subspace, represented as $|u\rangle \langle u|$. While generally, a density matrix is a positive semi-definite, Hermitian operator of trace one in a Hilbert space.

[12] Dirac notations are widely used in quantum probability, in which a *unit* vector $\vec{\mu}$ and its transpose $\vec{\mu}^T$ are denoted as a ket $|u\rangle$ and a bra $\langle u|$ respectively.

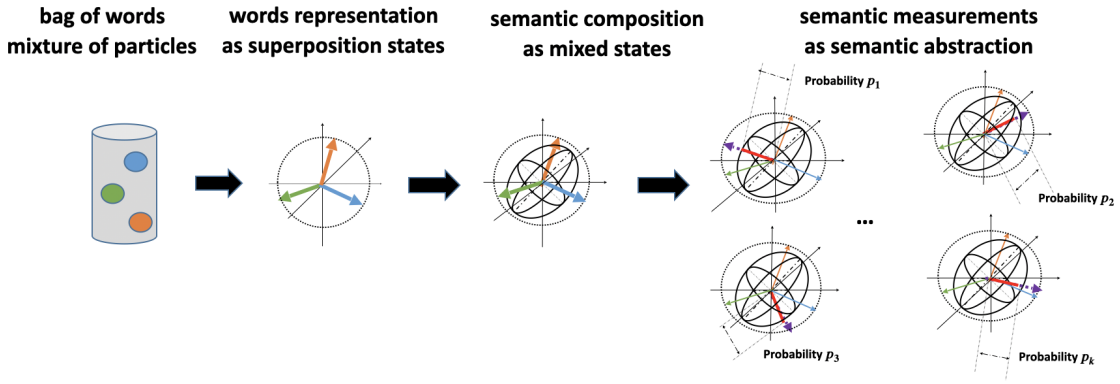


Figure 2.1: An intuitive analogy between natural language and physical particles.

Measures According to probability axioms [65], a probability measure $\mu(\cdot)$ defined on the Hilbert Space should meet 1) non-negative property: $\mu(|u\rangle\langle u|) \geq 0$ for any elementary event $|u\rangle\langle u|$; (2) unitary property: the probability for the entire sample space equals 1 $\sum_j \mu(|u_j\rangle\langle u_j|) = 1$ for any set of elementary events $\{|u_j\rangle\}$ that form an orthonormal basis.

Theorem 1 ***Gleason's Theorem****: For a quantum probability measure $\mu(\cdot)$ and a density matrix ρ, a probability for any event P is given by*

$$\mu_\rho(P) = tr(\rho P). \tag{2.7}$$

In particular, the probability of an elementary event represented by $|u\rangle\langle u|$ is $\mu_\rho(|u\rangle\langle u|) = tr(\rho\,|u\rangle\langle u|) = \langle u|\,\rho\,|u\rangle$.

By using Gleason's Theorem, the state of a single quantum system is represented by a density matrix ρ, which determines a probability measure on the space. The measured result of an elementary event is calculated by projective measurement using the event's projection operator.

2.3.2 Quantum formalization for natural language

This paper aims at modeling words as quantum particles, as intuitively shown in Tab. 2.1. 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 [16], [74], [86], [126]. The overview of the analogy is shown in Tab. 2.5. 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 a sentence is therefore modeled as particle mixing, resulting in 'a mixed system'.

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. (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

natural language	quantum theory
sememe	basis one-hot vector / **complete &orthogonal basis state** $\{\|e\rangle \,\|\, \|e\rangle \in \mathbb{R}^n, \|\, \|e\rangle \,|_2 = 1\}$
word	unit complex vector / **superposition state** $\{\|s\rangle \,\|\, \|s\rangle \in \mathbb{C}^n, \|\, \|s\rangle \,|_2 = 1\}$
semantic composition	density matrix / **mixed system** $\{\rho \| \rho = \rho^*, tr(\rho) = 1\}$
semantic abstract	projection measurement / **measurement** $\{\langle u\rangle\, u \| \|\, \|u\rangle \,|_2 = 1\}$
semantic representation	measured results / **measured probability** $\{\boldsymbol{p}\|\quad \|\boldsymbol{p}|_1 = 1, 0 < \boldsymbol{p}_i < 1\}$

Table 2.5: Analogy between natural language and concepts in quantum theory [134]

intrinsic uncertainties in natural language make quantum probability theory a desirable framework in this scenario.

2.3.3 Difference with existing works

2.3.3.1 The difference with previous Quantum-inspired works in IR/NLP

It is worth noting that using QPT in NLP is not new. A few works claimed it had preliminarily adopted QPT in NLP due to it adopted density matrices, the core component in QPT. [118] extends standard language model to model term dependency within a probabilistic framework of QT. [16] proposes the use of density matrix for semantic composition. There are many extended variants of the above works, for example, [144], [155].

In this work, we argue that the adoption of only density matrices (like the above) does not necessarily form a well-designed QPT. A probability theory should have a well-designed sample space and as well as probability measures. The above work and their following work do not adopt a probability measure defined in QPT, instead, they use some typical distance based measurement that does not result in probabilities, e.g., VN divergence in [118] and trace inner product in [16]. Our work is the first work to fully utilize QPT modeling in NLP that both the sample space and probability measurement in NLP scenarios is consistent with QPT.

2.3.3.2 The difference with probabilities in attention mechanisms

One may expect attention mechanisms [9] to achieve the aim of probabilistically driven networks. Although attention usually builds upon an embedding layer, it learns a probability distribution over individual tokens (by using the softmax activation), which goes back to assume that words are independent – even if they have been embedded as vectors and they are typically not orthogonal.

The reason is that the classical probability theory (that is derived from Kolmogorov axioms) is limited to treat mutually exclusive outcomes for events, or say, modeling words under "independence assumption". While quantum probability theory is directly defined in vector space, measurement is conducted by a subspace projection in vector space, which is not based on sets. We argue that quantum probability theory fits neural networks with

word embedding better, in which words are not necessarily independent, and still can be measured by subspace projections.

Plus, attention is a part of the whole network architecture, e.g., transformer with or an RNN/LSTM. Other than attention, other components are not probability-driven, for example, query/key/value transformation, feed-forward network, or RNN/LSTM cells. One network in which components are fully probability-driven, that is to say, has a concrete physical meaning corresponding to probability, i.e., states, measurements, measured probabilities, is expected.

The fundamental difference between the proposed probability-driven framework and softmax-activated attention is, the former is fully probability-driven with every meaningful component that conveys a concrete physical meaning corresponding to probabilities, i.e., states, measurements, measured probabilities; while in the latter, attention itself is partially probability-driven – only softmax-activated values form a probability distribution, but other components, e.g., query/key/value transformation, feed-forward network, or RNN/LSTM cells are not probability-driven.

2.3.3.3 The difference with the probabilistic interpretation of word embedding

[14] proposes a Neural Network Language Model (NNLM) that learns a language model that calculates the probability function of word sequences and predicts the next word.

$$P(w_{t+1}|w_1, w_2, \cdots w_t)$$

In practice, calculating such a probability distribution is nothing else but to activate the output of black-box neural networks with softmax functions. Let $f : V^t \longrightarrow R^V$ be the neural network or predict the $t+1$-th word using the previous t words, it learns with an objective:

$$\text{softmax}(f(w_1, w_2, \cdots w_t)) \propto P(w_{t+1}|w_1, w_2, \cdots w_t)$$

Probability distributions as hidden states using softmax activation in neural networks is arguably to make *a part of a neural network* more interpretable, since probabilities are normalized in a sense each of them ranges from 0 to 1 and their sums equal to 1. Most importantly, the hidden states could be probabilistically grounded into some specific events with respect to input words/tokens. Another example where probabilities help in terms of interpretation is the so-called 'attention mechanism'; many post-hoc visualizations were based on attention, which helps better understanding the working mechanism of neural networks in NLP [9], [127]. However, using softmax activation does not contribute to the *overall* interpretation of neural networks: neural networks are still black boxes. Some works [3], [4], [90] aim at understanding the intrinsic properties (e.g., similarity, paraphrases, word analogies, and geometry) of word vectors. Those works help the interpretation of word vectors, but they do not deal with the role of word vectors in neural networks. The challenge is to understand the unknown hidden states in neural networks when dense vectors are used as word representations. Our solution is to define all linguistic units, including words, in a unified Hilbert space in which words themselves could interpret other linguistic units.

2.4 Modeling Words as Waves for Sequential Modeling

2.4.1 Challenges to modeling sequence in vector space

Previous works investigate how to directly encode an additional sequential variable into a D-dimensional vector that has the same shape of the word vector:

$$g : \mathbb{N} \to \mathbb{R}^D \tag{2.8}$$

This is the case of *position embedding* in [41] and *time embedding* in [62].[13] Given g, the representation of w_i in its spatially or temporally sequential stamp t is

$$U_{i,t} = f_i + g_t \tag{2.9}$$

Typically, $t \in \mathbb{N}$ and $t \in [1, T]$. f is the word embedding and g is the sequential timestamped embedding. The physical meaning of $U_{i,t}$ could be the word representation of w_i in (a) in the t-th spatial position of a sentence/document or (b) in the t-th temporal time during semantic evolution.

Now we consider two issues of the above formalization: (1) dependency between words and the sequential variable, and (2) generalization to longer sequences.

2.4.1.1 Dependency between words and the sequential variable

By using Eq. 2.9 to combine word information and sequential information (i.e., position or time), it naturally assumes that word information is decoupled with sequential information. Let us first consider the two specific cases under decoupling scenario induced from Eq. 2.9.

Case 1: different words in a specific position/time. The difference between any two words is always constant concerning position/time:

$$U_{i,t} - U_{j,t} = (f_i + g_t) - (f_j + g_t) = f_i - f_j$$

The result is a constant with respect to position/time, formally,

Property 1. **Between-word relationship is translation invariant over positions/time**. $U_{i,t} - U_{j,t} = \mathcal{E}(i, j), \quad \forall t$

Case 2: Same word in different positions/time. For any word, the difference between its representation in t_1 and t_2 is constant.

$$U_{i,t_1} - U_{i,t_2} = (f_i + g_{t_1}) - (f_i + g_{t_2}) = g_{t_1} - g_{t_2}$$

The result is word-free, formally,

Property 2. **All words share the same trajectory over positions/time**. $U_{i,t_1} - U_{i,t_2} = \mathcal{E}(t1, t2), \quad \forall i$

Interestingly, both Property 1 and Property 2 fit the spatial case.

However, in the temporal case,

- Property 1 indicates that the relationship between words does not evolve over time; this is not always the case, for example, the relationship between *Trump* and *president* should be different in 2018 and in 2021.

[13] The time embedding proposed in that work is a "model-agnostic vector representation for time" in [62], not a temporal embedding for words.

- Property 2 indicates that all the words meaning should evolve according to the same trajectory; this is not always that case because some words may be "stable", some others gain a completely different meaning.

To overcome the above two issues in the temporal case, one has to design much more complicated interaction modules between a word component and a time component. Such a design may affect complexity, interpretability, or efficiency.

2.4.1.2 Generalization to longer sequences

To encode the position/time in vector space, one trivial approach is to learn T individual vectors that are independently trained. Although this approach performs well empirically, it is impossible to generalize to longer sequences, i.e., when $t > T$. Some work [62], [127] explore extending time/position vectors as functions over position/time. Suppose there exists a group of vectors $\{g(1), g(2), \cdots, g(T)\}$, denoted as G that is composed of T D-dimensional vectors, Each dimension of G will be a function that is only valid when $t = 1, 2, \cdots, T$, resulting in T functions.

To consider a longer sequence, especially $t \gg T$, we need to consider a boundedness property otherwise the time/position embedding may not convergence when t is big enough. A *boundedness* property is necessary to ensure that the position embedding can deal with the text of any length (*pos* could be large in a long document), or time embedding with very long intervals.

Property 3. **Boundedness**: The function over the variable position/time should be bounded, i.e. $\exists \delta \in \mathbb{R}^+, \forall \mathrm{t} \in \mathbb{N}, |g(\mathrm{t})| \leq \delta$.

To this end, existing work [127] usually consider sinusoidal functions. One of the intuitions behind this may be that sinusoidal functions are always bounded thanks to the periodical property. But a formal explanation of the necessity (instead of sufficiency) is needed, since other property (like Property 4 introduced in 4.1.2) also matters. This thesis will provide a principled definition for the desiderata of a sound position embedding and a principled interpretation of necessity for *sinusoidal* position embedding in Chapter 4.

2.4.2 Encoding sequences as waves

This subsection will discuss wave-like modeling for sequential modeling.

2.4.2.1 Formulation

Word-agnostic position/order embedding A word-agnostic order (e.g., position and time) embedding [62], [127] is defined as

$$f : \mathbb{R} \to \mathbb{R}^D \tag{2.10}$$

One may consider binary coding as shown in Fig. 2.2, 16 numbers $(0 - 15)$ are encoded as four-digit binary numbers, each digit is labeled in different colors. Observe that the last digit in red is a periodical sequence of $[0, 1]$ with a period of 2, the second last digit in blue is a periodical sequence of $[0, 0, 1, 1]$ with a period of 4, and so on.

One problem is that the above discrete binary coding is indifferentiable and it is therefore infeasible for backpropagation if the embedding is used in neural networks. To this end, one may consider designing continuous coding with the same periodical property. Fig. 2.2 shows a alternative sinusoidal encoding with periods of $2, 4, 8, 16$. Such continuity will facilitate backpropagation if such embedding is used in neural networks.

Binary coding: orders in 16 numbers $(0-15)$ are encoded as four-digit binary numbers 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, . Observe that the last digit in red is a periodical sequence of $[0, 1, \cdots]$ with a period of 2, the second last digit in blue is a periodical sequence of $[0, 0, 1, 1, \cdots]$ with a period of 4, and so on [14].

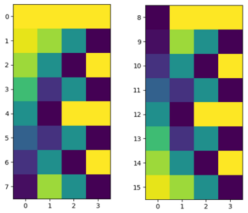


Figure 2.2: Sinusoidal coding for $0 - 15$

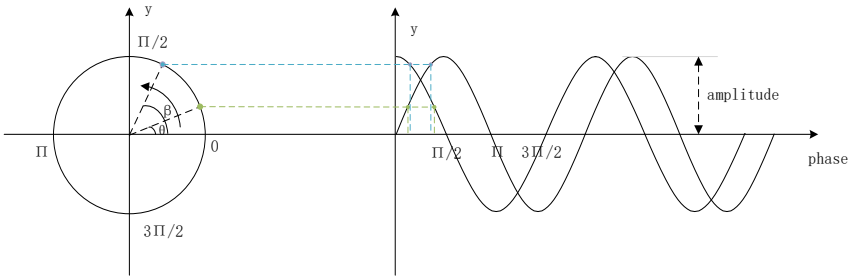


Figure 2.3: The order information generated by fixed amplitude and phase change, which is displayed on sine and cosine waves.

In practice, since the domain of order may be in a specific range, such periods (e.g. $[2, 4, 8, 16]$) would not necessarily be a geometric sequence. One can define exponentially-decay periods range from a pre-defined minimum and maximum period as below:

$$P = \min \times [(\frac{\max}{\min})^0; (\frac{\max}{\min})^{1/D}; (\frac{\max}{\min})^{2/D}; \cdots (\frac{\max}{\min})^1] \tag{2.11}$$

min is the smallest period and max is the largest period. For example, [127] choose a parameterization as $\min = 2\Pi$ and $\max = 20000\Pi$ [15]. Note that the encoding will nearly be a constant if the period is largely bigger than the maximum order (e.g., max sequence length is 512 or 1024 in Transformer [127]).

By doing so, each dimension of f is sinusoidal function with a variable t.

$$f_i = \sin(\omega_i t), \quad \omega_i = 2\Pi/P_i; 0 < i < n \tag{2.12}$$

t could be time in temporal dimension or position in spatial dimension. One can replace sin with cos or a combination of sin and cos.

By adding the sequence index, i.e., tt as a multiplier in the phase as in Eq. 2.12, this allows us to model word positions as waves — see Fig. 2.3. Such a parameterization will result in two important properties as stated in [137]: (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 previous works [127], [135]. In detail, suppose that we have a position encoding $f : \mathbb{R} \to \mathbb{R}^N$, there exists a transformation $g(k)$ that

[15] Namely, each of its frequencies are $\omega_i = (1/10000)^{2j/D}$

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, see the pooling strategies for feature maps after convolution in vision [69]. 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 the above statements needs to be further investigated, but the many existing works used such a position-free offset transformation and it was shown to empirically perform well – this is, for instance, the case of transformer architecture with sinusoidal position embeddings [127] which is the backbone of modern NLP.

Word-aware dynamic evolution Sequential encoding becomes more challenging when such sequential evolution is not word-agnostic, for example, an individual word may have a meaning which evolves over time and the change trends are not shared among words. In this scenario, modeling word-aware dynamic evolution is necessary.

word-aware dynamic evolution assumes that each word has a individual evolution which may vary with time/order. A word with index i has its meaning represented as a D-dimensional vector at a given time t. Therefore, word-aware dynamic evolution can be formalized as a mapping $f(i,t)$

$$f : \mathbb{N} \times \mathbb{R} \rightarrow \mathbb{R}^D \tag{2.13}$$

2.4.3 Spatial application: position-encoded word vectors

When processing text, the sequential structure of language is important, but it can be computationally costly to model with Neural Networks (NNs) [116] due to the difficulty in parallelization. This has been alleviated by modeling word sequence not at the NN architecture level, but by adding *position embeddings* at the feature level. This has been done by the convolutional sequence model (ConvSeq) [41] and the Transformer model [127]; the latter replaces recurrent and convolution operations with purely attention mechanisms. The next section will give an example of how Transformer cannot model position at the architecture level, thus evidencing the necessity to add position embeddings.

2.4.3.1 Permutation invariant issues in Transformer Architecture

A transformer layer consists of a self-attention (SAN) module and a feed-forward network (FFN) module. We will show that both modules are permutation invariant concerning word positions. An input X for SAN will be linearly transformed into query, key, value, and output space $\{Q, K, V\}$ as below:[16]

$$\begin{bmatrix} Q \\ K \\ V \end{bmatrix} = X \times \begin{bmatrix} \boldsymbol{W}^Q \\ \boldsymbol{W}^K \\ \boldsymbol{W}^V \end{bmatrix} \tag{2.14}$$

The self-attention mechanism (a.k.a Scaled Dot-Product Attention) is calculated as

$$\text{Attention(Q,K,V)} = \text{softmax}(\frac{QK}{\sqrt{d_k}})V \tag{2.15}$$

[16]For all linear transformation in this paper, the bias term is in default omitted

Let Π be a permutation for input tokens (or called 'words'), and Π^{-1} is its reverse. It can be easily checked that

$$\text{Attention}(\Pi(Q), \Pi(K), \Pi(V)) = \text{softmax}(\frac{\Pi(Q)\Pi(K)}{\sqrt{d_k}})\Pi(V) = \Pi^{-1}(\text{Attention(Q,K,V)}) \tag{2.16}$$

And the feed-forward neural networks are independently calculated token by token, which also does not consider the word order. More precisely, the overall Transformer output is in principle permutation invariant. This is unacceptable for language modeling.

Thus, transformer (including ConvSeq) cannot model the sequential order of words at the architecture level; a trivial way to address this issue is to empirically add extra position embedding at the feature level, by directly injecting word position in the word representations. Namely,

$$X = \text{WE}_x + \text{PE}_x \tag{2.17}$$

This assumes that word representations also depend on their absolute position appearing in a document. Position embedding is a practical approach to trade off the computing resource against the incorporation of linguistic structures.

More generally, vanilla position embeddings [41] assume that individual word positions are independent and do not consider relations between neighboring word positions. We posit that *both* the global absolute positions of words *and* their inner sequential and adjacent relationships are crucial in language. This is supported by recent empirical findings by [114] and [29] who show the importance of modeling distance between sequential elements and explicitly use extra relative position encodings to capture the relative-distance relationship of words. That is, the position embedding of a word should be more closed to its neighboring words than the word which is far from it. For example, the position embedding of a phrase (with more than one word) should not fundamentally change if it is moved from one position to another one, at least, its inner dependence should be preserved.

2.4.3.2 Problem definition

A *Word Embedding*(WE) generally defines a map $f_{we} : \mathbb{N} \to \mathbb{R}^D$ from a discrete **word** index to a D-dimensional real-valued vector and $\mathbb{N} = \{0, 1, 2, \ldots\}$. Similarly, a *Position Embedding* (PE) [41], [127] defines another map $f_{pe} : \mathbb{N} \to \mathbb{R}^D$ from a discrete **position** index to a vector. The final embedding for word w_j ($w_j \in \mathbb{W}$ with index j in a given vocabulary $\mathbb{W}$) in the *pos*-th position in a sentence is usually constructed by the sum

$$f(j, pos) = f_{we}(j) + f_{pe}(pos), \tag{2.18}$$

and $f(j, pos) \in \mathbb{R}^D$. Since both the word embedding map f_w and the position embedding map f_p only take integer values as word indexes or position indexes, embedding vectors for individual words or positions are trained independently. The independent training for each word vector is reasonable, since a word index is based on the order of a given arbitrary vocabulary and does not capture any specific sequential relationship with its neighboring words. However, the position index captures an ordered relationship, for instance, adjacency or precedence, leading to the problem that position embeddings in individual positions [41] are independent of each other; the ordered relationship between positions is not modeled. We refer to this as the *position independence problem*. This problem becomes more crucial when position embeddings are used in position-insensitive NNs, e.g., FastText [89], ConvSeq [41] and Transformer [127], because it is hard for such position-insensitive NNs with vanilla position embeddings [41] to infer that w_{j_1} in the

pos-th position is close to w_{j_2} in the $pos+1$-th position, or that w_{j_1} precedes w_{j_2}; instead, it is only inferred that w_{j_1} and w_{j_2} are in different positions, while the relative distance between them is almost unknown. Thus vanilla position embeddings [41] cannot fully capture the sequential aspect of language.

In Sec. 4.1 of this thesis, we will explain that the absolute positions of words are not informative while their relative distance matters. The reason is that absolute positions may not change too much the overall meaning of linguistic units like words/phrases/sentences while the order between them may change their overall meaning.

2.4.4 Temporal application: dynamic word embedding

Word meaning changes over time as a reflection of the changes of human society. Not only does word meaning change in a short time, but word meaning may also be subject to evolution over long timespans. The causes of the change of word meaning may be cultural, societal, or technological [121]; for example, the word `gay` shifted from the meaning '`cheerful`' in the 1900s to the meaning '`frolicsome`' in the 1950s and finally to the meaning '`homosexuality`' since the 1990s [47]. A word may even disappear because of the replacement with another word or a new word may come up because of new technology, situation, or phenomenon. The change of word meaning makes information representation, processing, and access difficult since a concept expressed by one word in a textual document might not match the same concept expressed by another word in another textual document, thus requiring models, methods, and tools to align word meanings and overcome the failures of computerized systems such as information retrieval systems.

The change of word meaning, a.k.a., lexical semantic change or diachronic word evolution, has been investigated for some decades [21], [121]. Recently, some surveys on this subject have been published; for example, an up-to-date survey of lexical semantic change using computational approaches has been published in [121], thus providing an overview of the complexity of models, methods, and tools available to address the issues of the change of word meaning.

Recent approaches for modeling the change of word meaning rely on distributed representations of words [14], [89] and on adopting time-specific word vectors as diachronic word representation. The training is based on diachronic text corpora, which are obtained by partitioning some text corpora into bins of textual documents labeled by time. When an approach relying on distributed representations of words is adopted, the training phase to learn diachronic word representation utilizes diachronic text corpora, which are obtained by partitioning some text corpora into bins of textual documents labeled by time.

2.4.4.1 Alignment Issues for Rotation-invariant Word Embeddings

There are typically two ways to obtain word vectors: one method is *count* models e.g., using matrix factorization [12]. While the other method is prediction-based neural methods. However, the former method is related to the latter; for example, Skip-gram with negative sampling can be described as factorizing an implicit matrix [71].

Consider matrix factorization. A common approach [148] is to approximate a given Positive Pointwise Mutual Information (PPMI) matrix $\boldsymbol{M}_t \in \mathbb{R}^{V \times V}$ by means of the V word vectors $\mathbf{U}_{\cdot,t}$ such that $\mathbf{U}_{\cdot,t}\mathbf{U}_{\cdot,t}^T \approx \boldsymbol{M}_t$. The entries of $\boldsymbol{M}_t$ are measures of a relationship between words; for example, the entry at row i and column j may measure the degree of co-occurrence of the i-th word with the j-word within fixed-size textual windows. The word embeddings can be found by either using an eigenvalue method or a matrix factorization

method which finds

$$\mathbf{U}_{\cdot,t} = \arg\min_{\boldsymbol{V}} |\boldsymbol{V}\boldsymbol{V}^T - \boldsymbol{M}_t|_F \tag{2.19}$$

However, we have that $\mathbf{U}'_{\cdot,t} = \mathbf{U}_{\cdot,t}\boldsymbol{R}$ can be an alternative solution of Eq. 2.19 for any orthogonal rotation matrix $\boldsymbol{R} \in \mathbb{R}^{D\times D}$. Indeed, $\mathbf{U}_{\cdot,t}\boldsymbol{R}(\mathbf{U}_{\cdot,t}\boldsymbol{R})^T = \mathbf{U}_{\cdot,t}\boldsymbol{R}\boldsymbol{R}^T(\mathbf{U}_{\cdot,t})^T = \mathbf{U}_{\cdot,t}(\mathbf{U}_{\cdot,t})^T$, since $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}$. Therefore, training word embeddings according to Eq. 2.19 is non-convex [11] and as a consequence a global minimum might not exist.

Moreover, if dynamic word embeddings are trained twice on the same time-specific corpus yet with different random seeds, it will result in different results [6]. It is also possible that two word vectors corresponding to different times, i.e., $\mathbf{U}_{\cdot,t_1}$ and $\mathbf{U}_{\cdot,t_2}$ cannot be compared because they might be arbitrarily rotated. One has to align these time-specific word embeddings. Because of the equivalence between prediction-based word embedding and matrix factorization [71], this rotation-invariant issue holds in prediction-based word embedding, e.g., Word2Vec, as well.

A popular training paradigm within such an approach is known as *train-and-align*: first, the embeddings are trained separately for each pre-grouped time-stamped corpus; then, the embeddings are aligned, e.g., by computing orthogonal projections [47], [66], vector initialization [64], temporal referencing [35], parameter regularizers [108], [148], aligned compass [34], or latent diffusion [11]. The methods above do not model word meaning evolution as a continuous process, but consider one-hop transformation between two pairwise adjacent timestamps.

2.4.4.2 Problem definition

Representing words in vector space, i.e., a static word embedding [89] $h : \mathbb{N} \to \mathbb{N}^D$, is a common practice. However, approaches like [89] cannot model word dynamics over time when considering time-dependent corpora (e.g., studying the evolution of cultural aspects). Diachronic word embedding [47] (sometimes called dynamic word embedding [11]) assumes that each word has a meaning which may vary with time. One word w_i with index i has its meaning represented as a D-dimensional vector $\mathbf{U}_{i,t} \in \mathbb{R}^D$ at a given time t. Therefore, dynamic word embedding can be formalized as a mapping from $\mathbb{N}\times\mathbb{N}$ to $\mathbb{R}^D$, i.e., from the pair (i,t) to a D-dimensional vector defined over the real field.

Existing methods for dynamic word embedding separately train embeddings for each time $t \in \mathbb{N}$. Note that even if we assume $t \in \mathbb{N}$, all formalization can be extended to $t \in \mathbb{R}$; in practice, the granularity between two consecutive times is fixed, e.g., one year, thus representing t in $\mathbb{N}$ is enough. The vectors that represent V words in time t are defined as $\mathbf{U}_{\cdot,t} \stackrel{def}{=} [\mathbf{U}_{1,t}, \mathbf{U}_{2,t}\cdots, \mathbf{U}_{V,t}] \in \mathbb{R}^{VD}$.

2.4.4.3 Existing works for dynamic word embedding

"One-hop change assumption" for semantic evolution Some of the previous works rely on "one-hop change assumption" for semantic evolution e.g., [11], [47], [64], [66], [108], [148]. The basic rationale of these works is to align word vectors in all time slices.

In [11], the basic inductive bias to learn dynamic embedding is to assume the probability that transform word embedding from t to $t+1$ as below (see Eq. 4 in [11]):

$$p(U_{\cdot,t+1}|U_{\cdot,t}) \propto \mathcal{N}(U_{\cdot,t}, \delta_t^2)\mathcal{N}(0, \delta_0^2) \tag{2.20}$$

where $\mathcal{N}$ is a Gaussian distribution. The former prior aims to make two consecutive time-specific word embedding being close; its variance δ_t^2 is related to the time difference between

t-th and $t+1$-th timestamps. While the latter prior aims to prevent the embedding vectors from growing very large. Note that $p(U_{\cdot,t+1}|U_{\cdot,t})$ is independent from either $p(U_{\cdot,t+2}|U_{\cdot,t+1})$ or $p(U_{\cdot,t}|U_{\cdot,t-1})$.

In [47], the goal is to find an orthogonal Procrustes R^t to align two consecutive learned time-specific embeddings as below (see Eq. 4 in [47]):

$$R^t = \underset{Q^TQ=I}{\operatorname{argmin}} ||U_{\cdot,t}Q - U_{\cdot,t+1}||_F \tag{2.21}$$

Note that $\{R^t\}_{t=1}^T$ are independent.

[148] proposed to learn dynamic word bedding by minimizing (see Eq. 5 in [148]):

$$\underset{U_{\cdot,1},U_{\cdot,2},\cdots,U_{\cdot,t}}{\text{minimizing}} \frac{1}{2}\sum_{t=1}^{T}||Y(t) - U_{\cdot,t}U_{\cdot,t}^T||_F^2 + \frac{\lambda}{2}\sum_{t=1}^{T}||U_{\cdot,t}||_F^2 + \frac{\tau}{2}\sum_{t=1}^{T}||U_{\cdot,t} - U_{\cdot,t-1}||_F^2 \tag{2.22}$$

$Y(t)$ is the PPMI matrix in time t. The only term to connect time-specific word vectors is the last one, namely, $\frac{\tau}{2}\sum_{t=1}^{T}||U_{\cdot,t} - U_{\cdot,t-1}||_F^2$ which also considers only two consecutive time-specific word embedding.

Words as functions A few words do not rely on a 'one-hop change assumption, e.g., [34], [35]. In [35] a temporal target embedding and a static context embedding are defined. A similar idea was proposed by [34] that aligns all time-specific source word vectors with a single compass in target embedding. Those works cannot model a "gradual" word meaning evolution.

The DiffTime Model proposed in [106] is the first that considers time as an independent continuous variable and thus defines words as functions. This, in principle, models evolution as continuous functions, and any real-valued variable of time is valid. In detail, in any training triplet with (u, v, t), t could be any real number. However, in practice, for simplicity, existing works tend to define time as integer numbers, by splitting a time span (1990-2016) as some time slices (1990, 1991,..., 2016) and the index of slices would be an identifier; these identifiers usually are integers in order, namely (1, 2,..., 17).

The DiffTime model first defines a time encoding by a two nonlinear layers using tanh activation functions as below:

$$f_{time}(t) = \tanh(M_2 \tanh(M_1 t + b_1) + b_2)$$

where $M_1 \in \mathbb{R}^{d\times 1}$ and $M_2 \in \mathbb{R}^{d\times d}$ are the weight terms and $b_1, b_2 \in \mathbb{R}^d$ are the bias terms.

A word encoder is represented as follows:

$$f_{word}(w_i) = T\vec{w_i} + B$$

where $T \in \mathbb{R}^{d\times d\times D}$ is a three-order tensor and $\vec{w_i} \in \mathbb{R}^{D\times 1}$ is the word vector of word w_i. By doing so, each word is then transformed to a matrix $f_{word}(w_i) \in \mathbb{R}^{d\times d}$

Finally, a word w in time t is presented as

$$U_{i,t} = M_4 f_{word}(w_i) f_{time}(t) + b_4$$

Note that the interaction operation between word encoder and time encoder is a matrix-vector multiplication.

One concern is if word functions learned in [106] could approximate any semantic

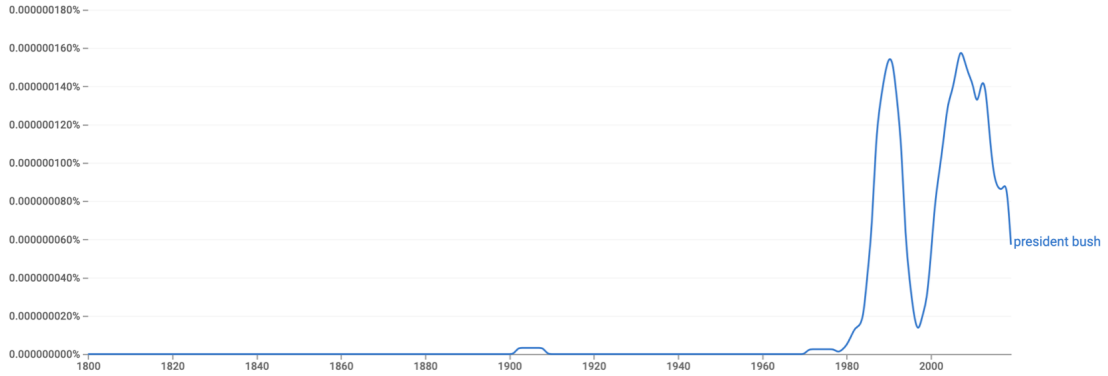


Figure 2.4: The frequencies of the phrase 'president bush' change over time. The figure is made from `https://books.google.com/ngrams/graph?content=president+bush`

evolution. Here we show a case where the semantic evolution may be complicated: in particular, we will focus on the word 'president'. The frequency of the phrase 'president bush' could to some extent reflect how the word meaning of 'president' is close to 'bush'. The PMI of two words is one metric of between-word relatedness and their frequency is one of the most important ingredients inside that is positive to PMI. PMI of two words is calculated as

$$\text{PMI}(w_i, w_j) = \log \frac{p(w_i, w_i)}{p(w_i)p(w_j)}$$

The term $p(w_i, w_i)$ is calculated by the co-occurrence frequency.

Figure 2.4 shows that the frequencies of the phrase 'president bush' change over time; from that figure one can observe that there are two peaks linked respectively to the presidency periods of the father (George H.W. Bush) and the son (George W. Bush). This example shows that the trends word functions should be able to capture may be highly nonlinear. The effectiveness of the DiffTime Model in this matter is not evidenced from a theoretical point of view.

Chapter 3

Words as Particles for Better Interpretation

Along with the power of neural networks, there is a growing concern about the interpretability of black-box Deep Neural Networks (DNNs); this concern is due to the impossibility of using predictions as a basis for decision making in many fields like health, commerce and law, unless we are able to justify those predictions in an explainable manner. Except for investigating the explainability of current DNNs, we could also build models bottom-up in an interpretable way. The post-hoc interpretability and transparency are two key requirements for an interpretable model developed by [77]. The former requires a model with explanations of why a model works successfully, while the latter concerns if the components of a model have self-explainability according to some mechanisms in the designing phase.

A possible solution to this problem is the adoption of the commonly-used attention mechanism. This mechanism adopts an explicit probability distribution among all input words and was arguably claimed to be more interpretable in neural networks [57], [112], [140]. However, the attention mechanism is limited to be used as an auxiliary component or a subcomponent of other network architectures like RNN [9] or Transformers [127]. This is because the classical probability theory derived from Kolmogorov axioms [65] does not fit well neural networks since it is set-based and hardly deals with events that are associated with word vectors instead of raw words.

Instead of using classical probability theory in the attention mechanism, in this thesis, we adopt a fully probability-driven neural network [74], [134] using Quantum Probability Theory (QPT) in which probabilities and their measurements are directly defined in vector space.

3.1 Quantum Probability Theory for Natural Language

Thus, a new probability theory is needed in vector based word representation. One can directly transform the original word based sample space to an abstract sample space, with each element corresponds to an abstract basis vector. The size of the sample space depends on the dimension of word vectors. To do so, a word, as an event, will not be one sample in sample space, but a weighted 'superposition' of many samples. A document with many words will therefore be a mixture of many superposed events. This complies with a quantum probability theory with such superposition and mixtures for particles.

Quantum probability (QP) theory is the basic mathematical framework of quantum physics that models uncertainty, which is a general tool to probabilistically describe any

Table 3.1: Physical meanings for transparency. DNN refers to typical Deep Neural Network.

Components	DNN	QPDN
Sememe	-	one-hot basis vector / **basis state**
Word	real vector	unit complex-valued vector / **superposition state**
N-gram	real vector	complex-valued density matrix / **mixed system**
Abstraction	CNN/RNN	unit complex-valued vector / **measurement**
High-level representation	real vector	probabilities/ **measured probability**

objects with uncertainly. Intuitively, we could treat a (textual) document as a physical system with multiple words (like particles). By doing so, we could potentially capture the polysemous aspect of words (like superposed particles) and their correlations/colocation (as entangled states with many particles). Also, the emerging research field of cognition suggests that there exist quantum-like phenomena in human cognition [1], especially language understanding [19], [20].

Thanks to the 'superposition principle', one can define a continuous relaxation of two mutually-exclusive events, which could considered as complex-valued interpolation of two mutually-exclusive events. Suppose that $|0\rangle$ and $|1\rangle$ are two mutually-exclusive events, one superposed state is defined as

$$|\Phi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{3.1}$$

α and β are complex numbers and $|\alpha|^2 + |\beta|^2 = 1$. Note that a new superposed state $|\Phi\rangle$ can neither be orthogonal to $|0\rangle$ or $|1\rangle$.

In this work, we adopt a probabilistic description (in Tab. 3.1) that originally describes micro particles e.g., photon, electron, etc. to probabilistically model words, N-gram, and documents. Intuitively, a polysemous word can naturally be represented as a superposed state while the documents can be considered as a mixed system with many words, in which the interaction between words may be implicitly encapsulated in an entangled connection like the connection between particles.

3.1.1 How it improve interpretability

In this work, transparency comes from the fact that the model is fully probability-driven, in a sense that each component in the neural networks corresponds to concrete physical meaning, for example, pure states (probability distribution over sememes) for words, mixed states (probability distribution over sememes) for multiple words, projection (probability measurements), and measured probabilities.

Especially, we formally reinterpret each dimension of word vectors as individual sememe in language; it, therefore, make the features, e.g., word vectors, themselves being individually interpretable. Based on the interpretable vector space spanned by sememes, we formally convey concrete meaning for all components as shown in Tab. 3.1. As a unified framework, sememes, words, and semantic abstraction are embedded in a unified Hilbert Space (a complex normed vector space) – thus neighboring words of each sememe could interpret the sememe and semantic abstraction. That is to say, we could provide some text (word) explanation for sememes and semantic abstraction (i.e., measurement subspace in our work) in a post-hoc manner, see Sec. 5.1.4.2 to know which we can see what we are exactly measuring. In contrast, it is nearly impossible for us to know what CNN kernels and RNN

cells do.

3.2 A Unified Framework for Linguistic Units

In this study, we model words and sentences as states in the same Hilbert Space $\mathcal{H}$ equipped with the probability function dictated by Gleason's theorem. To be consistent with the quantum world, the Dirac notation is adopted. Basically, a unit vector $\vec{\mu}$ and its transpose $\vec{\mu}^T$ are denoted as a ket $|u\rangle$ and a bra $\langle u|$ respectively.

3.2.1 Sememes as the basis Vectors

Sememes are the minimal non-separable semantic units of word meanings in language universals [43]. For example, the word 'blacksmith' is composed of sememes 'human', 'occupation', 'metal', and 'industrial'. We assume there are n semantic sememes $\{C_i\}_{i=1}^n$ in the text collection, which could be considered as a limited close set of sememes [17] in language universals [43]. The Hilbert Space $\mathcal{H}$ is then an n-dimensional finite space, where the concepts form a set of pure orthonormal states of the space, denoted as $\{|e_i\rangle\}_{i=1}^n$.

3.2.2 Words as superposed states

Each word w is modeled as a superposition state [94] in $\mathcal{H}^n$, admitting the following mathematical representation:

$$|w\rangle = \sum_{j=1}^{n} r_j e^{i\phi_j} |e_j\rangle \tag{3.2}$$

where $\{r_j\}_{j=1}^n$ are non-negative real-valued amplitudes with $\sum_{j=1}^n {r_j}^2 = 1$ and $\phi_j \in [-\pi, \pi], i = 1, 2, ..., n$ are the corresponding complex phases.

According to Eq. 3.2, $|w\rangle$ is a linear combination of the basis vectors $|e\rangle$ representing the basic concepts and weighted by the complex coefficients $z = re^{i\phi}$. According to Gleason's theorem [42], $|w\rangle$ is the state vector of w, whereas $|w\rangle \langle w|$ is the density matrix and $\langle e_j|w\rangle \langle w|e_j\rangle = |\langle e_j|w\rangle|^2 = |r_j|^2$ is the probability of e_k for the state (density) of $|w\rangle$.

It follows that the amplitude r_j is the squared root of a probabilistic weight of e_j. In NLP and IR, these probabilistic weights are usually estimated by counting low-level lexical features such as word co-occurrences. Therefore it is sensible to assume that r_j encodes low-level lexical features such as word co-occurrence information. On the contrary, we make the phases explicit and exploit them to encode another level of representation, that is, the semantics on a higher level as follows.

Distinct words such as "ivory" and "tower" have distinct meanings that vary depending on the context, thus enumerating a range of meanings. Indeed, a common English dictionary lists six different meanings of "tower" as a noun and five different meanings of "ivory" as a noun, thus suggesting at least thirty different meanings of "ivory tower", that is, 6×5 plus the meaning of the bigram. We correspond each meaning of a word w_k to one value of the phase ϕ_j in the range $[-\pi, +\pi]$, so that $r_j e^{i\phi'_j}$ differs from $r_k e^{i\phi''_j}$ for each $\phi'_j \neq \phi''_j$.

However, note that the probability of e_j given w is independent of ϕ_j, because the semantics at the higher level is a latent variable while only the manifest variables like co-occurrence frequencies can be observed and actually utilized to estimate the probabilistic weight. It is our aim to exploit and estimate the latent variables behind the semantics at a higher level. The situation can then be better visualized by Bloch's sphere than a flat, bidimensional Cartesian plane.

This is the first work to use complex-valued word representation in NLP/IR [73], [74], [134]. Previously, there was some related work to explore complex numbers in IR. For example, [158] proposed to use term frequency as the amplitude and encode Inverse Document Frequency (IDF) in the phase, which was empirically evidenced to be ineffective in practice. [86] explained the potential of complex values in IR. [118] used interference effects in order to model interactions between latent topics, which also involves complex numbers. To our knowledge, the work proposed in this thesis is the first work in NLP to link imaginary numbers in complex-valued representations to concrete meanings (i.e., word order), as introduced in Chapter 3.

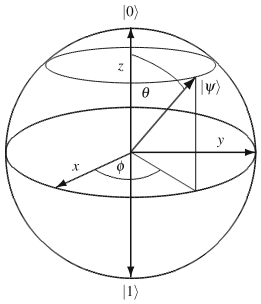


Figure 3.1: The figure provides an illustration of the Bloch sphere. This sphere is a visual representation of the space within which qubits live. The basic idea is that every qubit can be determined by only two angles, that is, θ and ϕ, as any geographical coordinate, can be determined by latitude and longitude [86]. The probability is given by the inclination of $|\psi\rangle$ only with respect to the vertical axis, and it is independent of the inclination of $|\psi\rangle$ with respect to the other axes.

Our approach can be seen as a generalization of the previous word embedding approaches [14], [88], [97], in a sense that it is a complex-valued embedding with unitary length as the constraint. On the other hand, such a complex embedding is fundamentally different from a double-length classical word embedding, in that it merges the different levels of information (i.e.., the semantic aspect of words, and higher-level aspect, polarity, ambiguity, or emotion) in a complicated way. When we apply any mathematical operations onto complex word embeddings, the resulting higher-level and lower-level information will always be a non-linear combination of both levels of information for all inputted words. Thus, the proposed complex word embedding increases the flexibility and representation power, making it possible to capture a more complicated combination of meanings that can be hardly represented by classical embeddings.

Suppose we enforce amplitudes to contain lexical features just like classical word embedding, while the phases contain alternative higher-level features such as the word polarity, word ambiguity, and its hidden emotion, just like what we assume in the introduction.

In the simplest case, two words w_1 and w_2 (or one dimension of two word vectors) are represented by two complex numbers $z_1 = r_1 e^{i\theta_1}$ and $z_2 = r_2 e^{i\theta_2}$. In an effort to compute their combined meaning, we perform a simple addition of z_1 and z_2, giving rise to $z = z_1 + z_2 = r^{i\theta}$. In the expression of z, the amplitude r and complex phase θ are a complicated, often non-linear combination of $\theta_1, \theta_2, r_1, r_2$: $r = \sqrt{r_1^2 + r_2^2 + 2r_1 r_2 \cos(\theta_2 - \theta_1)}$, $\theta = \arctan\left(\frac{r_1 \sin(\theta_1) + r_2 \sin(\theta_2)}{r_1 \cos(\theta_1) + r_2 \cos(\theta_2)}\right)$. In other words, both levels of information of a word combination are non-linear combinations of both levels of individual word features. When $\theta_1 = \theta_2 = 0$, however, it falls back to the classical case of linear combination. Complex-valued embedding, therefore, may be seen as a generalized embedding approach that

is naturally capable of representing the non-linear semantic composition of words. For instance, the word “hot dog" has a completely different meaning from “hot” and “dog”. While classical embedding approaches are likely to fail in this case, the new meaning could potentially be captured with complex word embedding. Of course, rather than a simple intuition, it would be interesting to see further investigations on the use of complex values, and a more comprehensive understanding of the way it links to the “meaning” of words.

3.2.3 Documents as mixed system

Based on the bag-of-words hypothesis [48], we formulate a sentence as a mixed system composed of the words it contains. Since each word is a superposition state in a Hilbert Space $\mathcal{H}^n$, a sentence is also viewed as a (mixed) state in the same space. By doing so we essentially ignore the order of words in the sentence. This assumption is adopted by most existing approaches.

Motivated by quantum probability [93], we further assume a sentence is a mixed state of semantic concepts, represented by a n-by-n density matrix ρ in the Hilbert Space $\mathcal{H}^n$. As a mixed system of word states, the density matrix ρ is computed as

$$\rho = \sum_{j}^{m} \frac{1}{m} |w_i\rangle \langle w_i|, \tag{3.3}$$

where m is the length of the sentence and each element of the density matrix a complex-valued. Observe that a word could occur multiple times. In particular, $|w_i\rangle \langle w_i|$ could be a density matrix depicting the pure state of the i^{th} word in the sentence.

The density matrix is interpreted as a semantic composition of words. It describes the distribution over the set of semantic concepts of the sentence, but in a non-classical manner. If all the off-diagonal elements are zero, it degenerates to a standard probability distribution over the basic state $\{|e_i\rangle_{i=1}^n\}$. More generally, the non-zero off-diagonal elements describe the complicated non-linear relationships between representation levels, which could play a crucial role in determining the emerging meanings when words are combined.

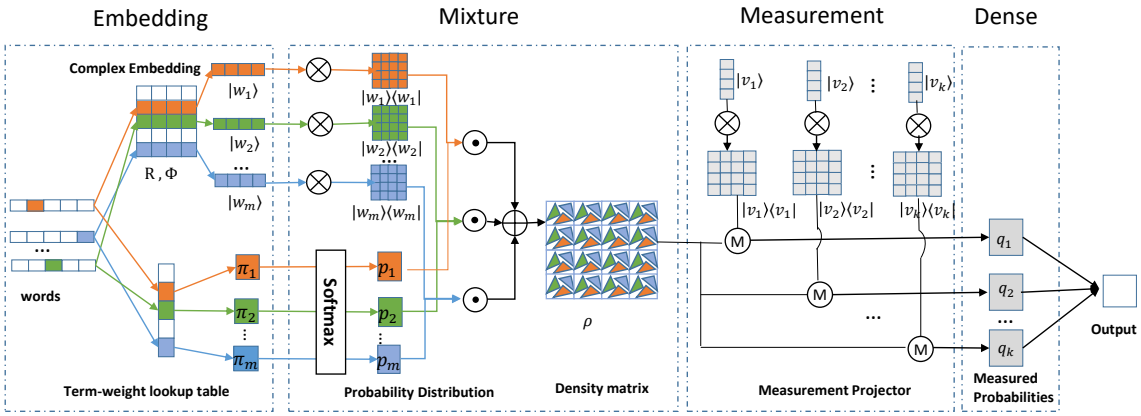
3.2.4 Measurements as semantic abstraction

A sentence is modeled as a mixed system and represented by a density matrix. Essentially, a density matrix determines a probability for any states in the Hilbert Space, so it contains a rich source of information. We then need to characterize the density matrix in such a way to better capture the relevant information concerning a task at hand.

To do so, we propose an efficient approach to extract information from the mixed state of sentence inspired by Quantum Tomography [55], [83], [102], which is the technique of characterizing a quantum state by applying a large number of measurements to the system, each time preparing the system anew. Based on the measurements and corresponding measurement results, Quantum Tomography provides different computational methods to reconstruct the state.

We believe that the measurements and corresponding measurement results can serve as a good characterization of a density matrix. Similar to quantum tomography, a set of trainable measurement projectors $\{P_i\}_{i=1}^k$ is applied to the mixed state of the sentence, and each time we prepare the quantum state using the same configurations. Different from quantum mechanics where it is impossible to produce a perfect copy of a quantum state [142], we can produce a replicate of the sentence mixed state for every measurement.

Figure 3.2: Architecture of Quantum probability-driven Neural Network [134]. $\odot$ means that a matrix multiplies a number with each elements. $\bigoplus$ refers to a element-wise addition. $\bigotimes$ denotes a outer production to a vector, $\textcircled{m}$ means a measurement operation according to Eq. 3.4.



According Born's rule [18] and Gleason's theorem [42], applying the measurement projector P_i onto the sentence density matrix ρ representing the sentence mixed state yields the following result:

$$p_i = tr(P_i \rho), \tag{3.4}$$

where $tr(\cdot)$ is the matrix trace, and each operator P_i uniquely corresponds to a pure quantum state or a unit-length complex-valued vector $|v_i\rangle$: $P_i = |v_i\rangle \langle v_i|$, and we call v_i a *measurement state*. The resulting measurement probabilities $\{p_i\}_{i=1}^{k}$ are used to as a group of high-level features for downstream tasks.

Different from quantum tomography, we do not impose additional constraints on the measurements, which means only part of the information can be extracted by the measurements. In practical applications, this may help to extract the crucial features whilst throwing away the less important information with respect to the given task. In this work, we adopt a flexible data-driven approach to identify the most discriminative measurements for determining the sentence class.

The trainable measurement projectors can also be understood from the perspective of supervised dimensionality reduction. Linear Discriminant Analysis (LDA) [38] tried to find discriminative projection directions for a better division of different classes. Similarly, the goal of trainable measurements is to find a group of finite measurement projectors to distinguish the possible labels, but in a sound quantum probability framework with complex values.

3.2.5 A united framework

To this end, this paper addresses the problem of interpretability by inspiring from QPT [128], which is the mathematical tool for describing quantum phenomena. Essentially, we seek to interpret semantic units of different levels as microscopic particles that are in states of interference or mixture, in an effort to formulate quantum-like phenomena inherent in human language [1], [20]. Apart from being interpretable, our model is capable of formulating complicated word semantic combinations through complex-valued word embeddings.

First, compared to the traditional network like CNN kernels and RNN cells, the proposed network is more interpretable in that each component can be interpreted as a quantum

concept with a concrete physical meaning (as shown in Tab. 3.1). A transparent bottom-up architecture is built to compose the semantic representation of the whole sentence from the basic sememes, words, and N-grams, in a single vector space. In particular, the *semantic measurements*, which correspond to specific measurement directions (like words), are also embedded in this space, thus making the learned measurements easily understood by a human.

An end-to-end neural network is proposed to compose lower-level semantic units to higher-level units in a bottom-up way to construct the internal representation. The parameters of the proposed model are $\Theta = \{R, \Phi, \Pi, \{|v_i\rangle\}_{i=1}^{k}, W\}$, denoting the amplitude, phase, and weight lookup table, the set of measurement states, and the weights for the dense layer respectively.

They are trained in the procedure shown in Algo. 1. In the proposed end-to-end network, these parameters can be updated by the back-propagation algorithm, with cross-entropy as the loss function. As shown in Algorithm 1, $\rightarrow$ means that it is a normal real-valued vector, while it should be a unit complex-valued one with Dirac notations like $\langle|$ or $|\rangle$. In order to obtain a better empirical result, we set trained weights for each word (like $\{u_i{}_{i=1}^{m}\}$ in Algo. 1), which is inspired by the IR literature, e.g., the IDF [119] term.

ALGORITHM 1: Training of QPDN

Result: Obtaining $\Theta = \{R, \Phi, \Pi, \{|v_i\rangle\}_{i=1}^{k}, W\}$
Initializing phase embedding Φ;
Initializing amplitude embedding R;
Initializing word weight embedding Π;
Initializing projector vector set $\{|v_i\rangle\}_{i=1}^{k}$;
Initializing weight W in dense layer;
while *not converge* **do**
  obtaining $\{|w_i\rangle\}_{i=1}^{m}$ from R and Φ according to Eq. 3.2;
  building density matrix $\rho = \sum_i^m p_i |w_i\rangle \langle w_i|$;
  measured result $\overrightarrow{q} = [q_1, q_2, ...q_k]$, $q_j = tr(\rho |v_j\rangle \langle v_j|)$;
  computing class label $\widehat{\overrightarrow{y}} = softmax(\overrightarrow{q} \cdot W)$;
  back propagating Θ with loss $cross\_entropy(\widehat{\overrightarrow{y}}, \overrightarrow{y})$;
end

Crucial to the back-propagation scheme are the gradients with respect to the sentence density matrix ρ and set of measurement $\{|v_i\rangle\}_{i=1}^{k}$. Let $E = cross\_entropy(\widehat{\overrightarrow{y}}, \overrightarrow{y})$ denote the loss function. The gradients $\frac{\partial E}{\partial |v_i\rangle}$ and $\frac{\partial E}{\partial \rho}$ are computed as follows:

$$\begin{aligned} \frac{\partial E}{\partial |v_i\rangle} = \frac{\partial E}{\partial q_i}\frac{\partial q_i}{|v_i\rangle} &= \frac{\partial E}{\partial q_i}\frac{\partial tr(\rho |v_i\rangle \langle v_i|)}{\partial |v_i\rangle} \\ = \sum_{i=1}^{k}\frac{\partial E}{\partial q_i}\frac{\partial \langle v_i| \rho |v_i\rangle}{\partial |v_i\rangle} &= 2\sum_{i=1}^{k}\frac{\partial E}{\partial q_i}\langle v_i| \rho, \end{aligned} \tag{3.5}$$

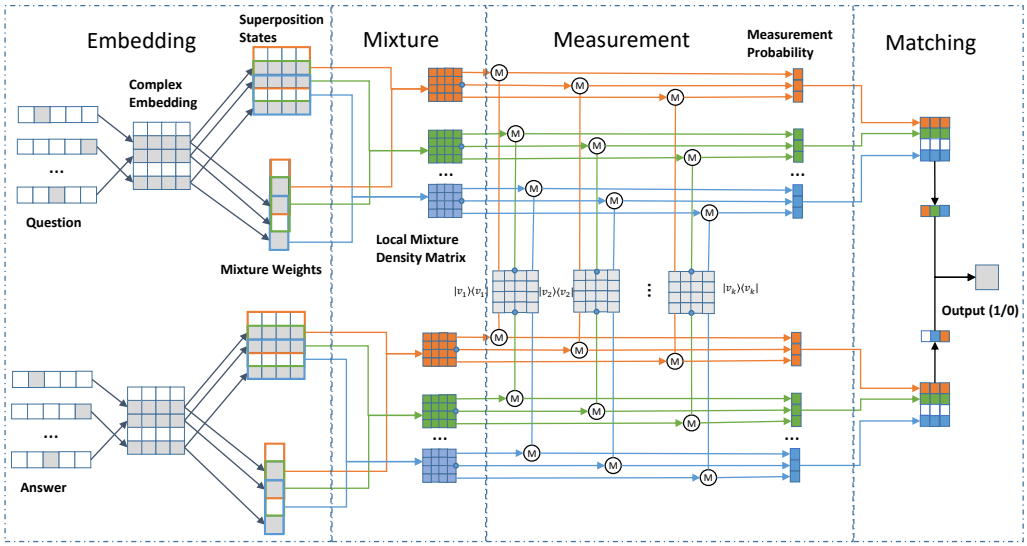


Figure 3.3: Architecture of Complex-valued Network for Matching. Ⓜ means a measurement operation according to Eq. 2.7.

$$\begin{aligned} \frac{\partial E}{\partial \rho} = \sum_{i=1}^{k} \frac{\partial E}{\partial q_i} \frac{\partial q_i}{\partial \rho} &= \sum_{i=1}^{k} \frac{\partial E}{\partial q_i} \frac{\partial \langle v_i | \rho | v_i \rangle}{\partial \rho} \\ = \sum_{i=1}^{k} \frac{\partial E}{\partial q_i} \langle v_i | v_i \rangle &= \sum_{i=1}^{k} \frac{\partial E}{\partial q_i}, \end{aligned} \tag{3.6}$$

where $\frac{\partial E}{\partial q_i}$ is simply the gradient for the dense layer input. Based on $\frac{\partial E}{\partial |v_i\rangle}$ and $\frac{\partial E}{\partial \rho}$, the gradients for all parameters in Θ can be computed following the classic back-propagation algorithm.

3.2.6 On complex-valued word embedding

Apart from the quantum representation in Eq. 3.2, a word w can also be represented as a complex vector in unit length $|w\rangle = [r_1 e^{i\phi_1}, r_2 e^{i\phi_2} ... r_n e^{i\phi_n}]^T$. Moreover, words, n-grams, and measurements adopt a complex-valued representation in a complex-valued vector space with sememes as the basis. On top of the complex-valued vector representation of words, operations like additions or projections naturally admit a non-linear word semantic composition. For example, a complex number z can be represented in amplitude-phase form, i.e. $z = re^{i\phi}$. Adding two complex vectors will result in a complicated non-linear combination of amplitudes and phases, which is a generalized but fundamentally different combination compared to real-valued addition. This thesis argues that the amplitudes contain lexical features like classical word embedding, while the phases contain alternative higher-level features such as the word polarity, word ambiguity, and its hidden emotion.

3.3 Extension to Text Matching

QPDN which was originally designed for single document representation, could be extended to match text pairs (e.g., a question sentence and an answer sentence), see the network architecture in Tab. 3.3. This network architecture is called 'Complex-valued Matching

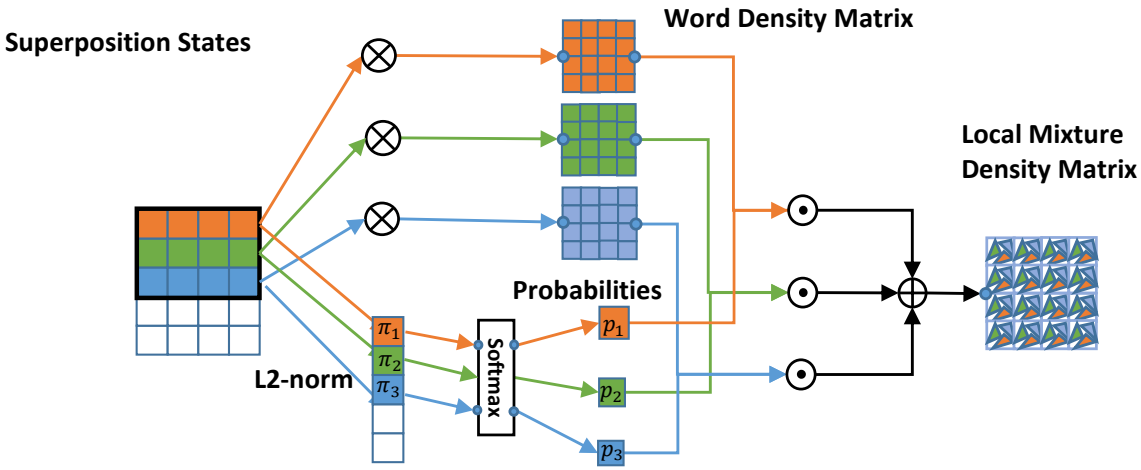


Figure 3.4: Architecture of local mixture component. $\odot$ means that a matrix multiplies a number with each elements. $\otimes$ denotes an outer product of a vector.

Network' (CNM), which was published in [74]. The main differences are (1) an extra local mixture scheme to extract features for each N-gram in two sentences in Sec. 3.3.1 and (2) an interaction module to match the two sentence representations in sec. 3.3.2.

3.3.1 Local mixture scheme

The sentence representation is modeled as a combination of individual word representations in the sentence. NNQLM proposed to model a sentence as a *global mixture* of all words, totally ignoring the word order information. We have improved this work by considering a local mixture of words, with the assumption that near words may be more semantically close to each other. As shown in Fig. 3.4, the proposed local mixture scheme adopts some sliding windows of length l (e.g. 3) to build multiple bottom-up density matrices for both question and answer sentences, resulting in a sequence of density matrices n-gram terms.

As for the semantic composition of n-grams, we propose an improved approach over Eq. 2.6, which implicitly assumes that each word state has the same weight. Uniform term weight is a rough estimation that does not hold in the empirical point of view. In the most commonly used TF-IDF weighting scheme, one of the components is the term IDF. In this study, we take the $L2$-norm length of the word vector as the relative weight in a local context window for a specific word, which could be updated during training. $L2$-norm is a measure of the semantic richness of a word, i.e. the longer the vector the richer the meaning. The density matrix of a sentence is computed as follows:

$$\rho = \sum_{i}^{l} p(w_i) \, |w_i\rangle \, \langle w_i|, \tag{3.7}$$

where the relative importance of each word $p(w_i)$ in an n-gram is determined by the word-dependent weight $\pi(w_i)$. In particular, we associate it with a probability value by passing the word-dependent weights through a softmax operation, $p(w_i) = \frac{e^{\pi(w_i)}}{\sum_j^l e^{\pi(w_j)}}$. This guarantees $\sum_i^l p(w_i) = 1$ and $tr(\rho) = 1$. Our word-specific weight is not a static value like IDF, but depends on the other words in the context during training.

Through this local mixture scheme, a sentence is represented as a sequence of complex-

valued density matrices as opposed to a single real matrix in NNQLM [155]. Moreover, the local mixture is a parameter-free component, which avoids the use of parameter-abundant layers and makes the network training more efficient.

3.3.2 Learning to match sentence pairs

We are inspired by the principle of quantum tomography [55], [83], [102], which essentially tries to conduct a series of measurements onto an unknown state and use the measurement results to reconstruct the unknown state. In this work, we introduce trainable measurements as a good characterization of the mixed system in density matrix form.

For a density matrix, we apply to them the same set of semantic measurement operators $\{|v_j\rangle\}_{j=1}^{k}$ via Gleason's Theorem – see Theorem 1.

$$p_{v_j} = tr(\rho \left|v_j\right\rangle \left\langle v_j\right|) \tag{3.8}$$

As mentioned before, $|v_j\rangle$ is an arbitrary unit complex-valued vector like a word. After applying these two measurements to ρ_1 for question n-gram and ρ_2 for answer n-gram, a set of probability values $\{p_{1j}\}_{j=1}^{k}$ and $\{p_{2j}\}_{j=1}^{k}$ are produced.

After the same measurements with multiple n-gram terms between a QA pair, we could a get a series probabilities for question and answer sentence respectively, in which each probability could also be considered as high-level feature extraction from multiple views of projectors, in order to characterize the sentence semantics in a topic level of granularity. Finally, we could get k-by-L (L is the sentence length) measured probabilities for both a question and an answer, and vector-based distances are applied onto the matrices to obtain the matching scores of QA pairs. Compared to directly calculate the distance between two density matrices, we turn to calculate the probability-based abstraction from the matrices, namely the measured probabilities $\{p_{1j}\}_{j=1}^{k}$ and $\{p_{2j}\}_{j=1}^{k}$, in which each element os a probability value ranging from 0 to 1. If a sufficient number of measurement states are applied, the distance of the output measurement probabilities could approximate an optimal distance estimator.

In [155] the trace inner product of two density matrices was used: $d(\rho_a, \rho_b) = tr(\rho_a \rho_b)$. However, this is not a theoretically sound metric. Since both $tr(\rho_a \rho_b) > tr(\rho_a^2)$ and $tr(\rho_a \rho_b) < tr(\rho_a^2)$ can happen for $\rho_a \neq \rho_b$, the distance value of $d(\rho_a, \rho_b)$ does not attain its extrema at $\rho_a = \rho_b$, which disagrees with our intuition on the measure of distance. The other model proposed by [155] uses CNN on the product of two density matrices $\rho_a \rho_b$, which loses the property of density matrix as a probability distribution. [94] introduced three measures namely trace distance, fidelity and VN-divergence. However, it is computationally costly to compute these metrics and propagate the loss in an end-to-end training framework.

We propose a new approach with trainable measurements to replace directly defining a metric between two density matrices. Moreover, the trainable measurements make our model more interpretable. Just like a word, each measurement vector is a superposition state in the Hilbert Space, which can be easily optimized by neural networks and understood by humans. From the perspective of linear discriminant analysis (LDA) [38], this approach is intended to find a group of finite discriminative projection directions for a better division of different classes, but in a more sound framework inspired by quantum probability with complex-valued values. From an empirical point of view, the data-driven measurements could have a better discriminative ability to classify whether the question sentence and the answer sentence matches or not.

Chapter 4

Words as Waves for Sequential Modeling

Word vectors are trained using co-occurrence information. Especially, in one of the most popular word vector methods called Glove [97], a dot product between two word vectors was used to approximate the co-occurrence count of such two words. [71] claimed that another popular word vector method called Word2Vec [89] implicitly factorizes a context matrix (PMI matrix) which is also related to cooccurrence.

However, other than cooccurrence, word representation should also involve other aspects, e.g. its positional order, time-aware semantic evolution, sentiment polarity, compositionality in a context. Among these aspects, positional order and the time aspect are nontrivial ones that this thesis focuses on.

4.1 Spatial Case: Position Encoding

From a reductionism point of view, any complicated sentence or document could be interpreted as a combination of its parts e.g., characters, words, phrases, clauses, etc. Currently, many advanced Neural Networks (e.g. CNN, RNN, Transformer) aim to represent a sentence/document by ensembling its inside words/subwords in a bottom-up manner.[1] The nontrivial point is that these words/subwords in a sentence/document are sequential instead of unordered. This section will formally define two general position-aware inductive biases [2], i.e., *Absolute Position Bias* and *Local Receptive Field*, in this bottom-up manner. Note that such biases are not associated with any specific word but they are general trends for individual positions.

Absolute Position Bias Words are processed sequentially by a human, and some of them may be nonuniformly distributed in individual positions due to syntax, semantic or pragmatic habits. Such nonuniform distribution on positions may lead to a possibility that some words in specific positions may statistically contribute more to the overall document meaning. *Absolute position* of a word in a sentence (or a document) matters since it is assumed that the word distribution in individual absolute positions varies. This thesis

[1] For example, in transformer architecture, a weighted sum of low-layer vectors is used as a high-layer input.

[2] See [44] for more general discussions about inductive biases in deep learning.

define a word distribution in specific position x as below:

$$p(x) \stackrel{def}{=} [\#(w_1, x), \#(w_2, x), \cdots, \#(w_V, x)] \tag{4.1}$$

$\#(w_i, x)$ denotes the count when a word w_i appears in x-th position in a sentence (a document) in whole training samples. $APB(x)$ corresponds to the word distribution $p(x)$ on position x. By taking APB into account, it, therefore, admits that *a word representation might have minor differences depending on where it appears in a document (or in a local segment).*

For example, some "interrogative words" (i.e., "how", "when", "where", "who", and etc.) are more likely to be the beginning of a document (let us say, $x = 0$). Since interrogative words may determine the question types such as a where-type or who-type question, models may need to pay more attention to these words to facilitate answer selections. Since "how", "when", "where", and "who" are more likely to be in 0-th position, the 0-th position embedding could express, at least to some extent, the question types.

Local Receptive Field Generally, some neural network blocks (denoted as f_{nn}) in NLP aim to transform a sequence of low-level word representations to high-level ones. For a document with n words, their $(l+1)$-layer word representations are based on the word representations of the previous layers. That is

$$[z_1^{l+1}, z_2^{l+1}, \cdots, z_n^{l+1}] = f_{\text{nn}}(z_1^l, z_2^l, \cdots, z_n^l) \tag{4.2}$$

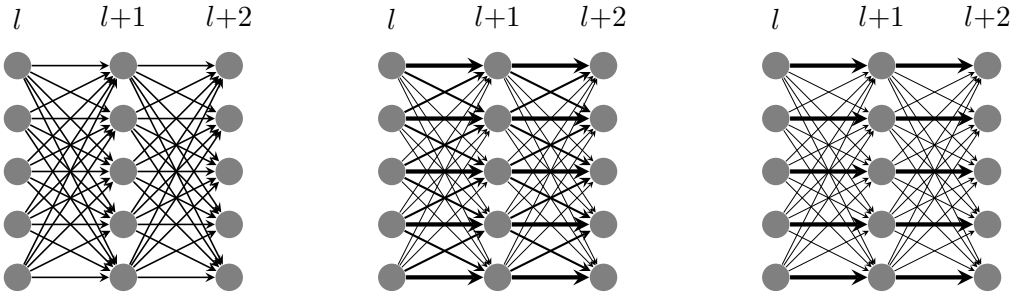
In Transformer, f_{nn} could be a self-attention module, a feed-forward network or a pipeline including the both. One can also treat a convolution neural network layer [41] as f_{nn}. In any case, f_{nn} will be linear or non-linear transformation. We consider the contribution amount (or called 'aggregation weight') of the low-level word representations to high-level word representations as p. $p_{i,j}^l$ measures to which extent the z_i^l depends on $z_j^{(l+1)}$. p could reflect the general patterns for semantic aggregations.

There are typically three aggregation paradigms to ensemble lower-lever semantic units (e.g., subwords or words) to higher-level semantic units (e.g., clauses or documents). They are shown in Figure 4.1. The Data flow diagram shows how low-level word representations interact to be aggregated in higher-level word representations. *Fully-attended data* flow shows that semantic aggregation is based on the bag-of-word assumption that does not consider word order – see a Transformer model without position embeddings – namely, $p_{i,\cdot}^l$ is uniform. *Fully-attended data* flow shows that semantic aggregation will pay more attention on the local words – see a typical Transformer or Convolution – namely, $p_{i,j}^l$ is monotonically decreasing to $|i-j|$. *Self-attended data flow* shows that there should be a extra bias for semantic aggregation to concentrate more on the center semantic unit – see a Transformer with random position embedding – namely, namely, $p_{i,i}^l > p_{i,i}^l$ if $i \neq j$.[3]

In general, the fully-attended aggregation focuses globally on long-term dependencies like Multiple-layer Perception (MLP); locally-attended aggregation tends to accumulate local neighboring semantic units to a global one; self-attended aggregation pays more attention to semantic units themselves (like each segment in BERT tends to attend in-

[3]The position embeddings are claimed to impose a position-aware attention bias when calculating attention values in Transformer [29] With position embedding, the position-by-position bias matrix is calculated by the outer product of position embeddings $\mathbf{P} \in \mathbb{R}^{L \times D}$, namely, $\mathbf{B} = \mathbf{P}\mathbf{P}^T \in \mathbb{R}^{L \times L}$. When position embeddings are random, $\mathbf{B}$ will be a matrix that the diagonal elements are significantly bigger than off-diagonal elements, since a dot product between a vector and itself is more likely bigger than the counterpart between a vector and another random vector.

segment words). The first and last one may, to some extent, lose the ordered information between input units, leading to lower expressiveness.



(a) fully-attended data flow. (b) locally-attended data flow (c) self-attended data flow.

Figure 4.1: The three typical data flows for semantic aggregation. A circle is a set of neurons (a.k.a, a hidden state) that represent a word. An arrow refers to a aggregation weight from a low-level word representation to a high-level word representation, and its thickness reflects the amount of the weight.

For language, neighboring words are more likely to constitute together a higher-level semantic unit" than far-way words with long-term dependencies. Capturing the relative distance between words is crucial to implement the locally-attended aggregation. In this thesis, we will refer to this neighbor aggregation as "Relative Position Bias".

The position index captures an ordered relationship, for instance, adjacency or precedence, leading to the problem that position embeddings in individual positions [41] are independent of each other. It is hard for NNs with vanilla position embeddings [41] to infer that w_{j_1} in the pos-th position is close to w_{j_2} in the $pos+1$-th position, or that w_{j_1} precedes w_{j_2}; instead, it is only inferred that w_{j_1} and w_{j_2} are in different positions, while the relative distance between them is almost unknown. Thus vanilla position embeddings [41] cannot fully capture the sequential aspect of language.

From a biological point of view, one may not gaze at an entire article in a glance. Most language designs smaller self-contained units with smaller granularity, like paragraphs or sentences. Moreover, since the pupil's annotation range is limited to a few words, the way humans processes words may more likely be a local manner, or called 'a local field' before. The interaction between neighborhood words matters. Such the amount of interaction should decrease with a longer relative distance between words.

On the other hand, linguistic units like words, phrases, sentences should not highly depend on their absolute positions. This not only simplifies the language complexity and also facilitates human reading no matter whether they begin. The thesis here also make an assumption that

> ***Absolute/relative position assumption****: absolute position brings little information for words while the relative distance between words matters.*

The above assumption will induce a so-called 'Position-free offset transformation' property in Sec 4.1.2.

4.1.1 Extending word vectors to word functions

The vanilla fully-learnable position embeddings do not consider the ordered relationship between positions, since the position embeddings are indexed separately. This thesis

proposes to build continuous functions over a variable (i.e., position index) to represent words. Formally, we define a general embedding as

$$f(j, \text{pos}) = \boldsymbol{g}_j(\text{pos}) \in \mathbb{R}^D, \tag{4.3}$$

where $\boldsymbol{g}_j$ is short for $\boldsymbol{g}_{we}(j) \in (\mathcal{F})^D$, indicating D functions over position index pos, and $g_{we}(\cdot) : \mathbb{N} \to (\mathcal{F})^D$ is a mapping from a word index to D functions. By expanding the D dimension of $\boldsymbol{g}_j$, a word w_j in the *pos*-th position can be represented as a D-dimensional vector as shown in

$$[g_{j,1}(\text{pos}), g_{j,2}(\text{pos}), ..., g_{j,D}(\text{pos})] \in \mathbb{R}^D, \tag{4.4}$$

in which $\forall g_{j,d}(\cdot) \in \mathcal{F} : \mathbb{N} \to \mathbb{R}, d \in \{1, 2, ..., D\}$ is a function over the position index *pos*. To move the word w_j from the current position *pos* to another one *pos*′, it needs only replace the variable *pos* to *pos*′ without changing $\boldsymbol{g}_j$.

Functions for words, especially continuous functions, are expected to capture smooth transformation from a position to its adjacent position, therefore, modeling word order. The *position-independent* position embedding [41] can be considered as a special case of our definition when it only takes independent values for individual positions in the embedding function. In such an embedding function, it is only valid when positions are in a limited set of pre-defined positions and the function is not continuous and values are independent of each other, e.g., adjacency position vectors are not related.

These **continuous** functions will help when positions are not necessarily integers. For example, this may help when we want to insert some information between two successive phrases/words like in [79], where the goal was injecting involved knowledge to a given sentence, which adds an extra 'clause' as the input of BERT and builds a visible matrix.[4] Position embeddings defined in real-valued space could help to jointly model the extracted positions and the inserted pseudo positions without using an extra visible matrix.

4.1.2 Desiderata

Relative distance is hard to compute because position indices are not visible in NNs after vector embedding (discrete position indices are necessarily embedded as vectors like words to be back-propagated with the gradient). Hence, we claim that the modeling of relative distance in NNs should be *position-free*: absolute position indices cannot be directly accessed in intermediate layers. Instead of processing *position-free* operations in NNs to capture the relative distance between words, prior work [29], [114] first calculates the relative distance between words, and then feeds the relative distance as an additional feature or as embeddings/weights to NNs, instead of directly feeding with the raw position indices.

Assume that words are embedded into $\mathbb{R}^D$, and let, for $1 \leq d \leq D$, the function $g_{j,d} : \mathbb{N} \to \mathbb{R}$ be the embedding function giving the d-th coordinate of the representation of word w_j (i.e., $g_{j,d}(\text{pos})$ is the d-th coordinate of the embedding of w_j if it occurs at position pos. In the following, we simply write g instead of $g_{j,d}$ when there is no risk of confusion. Ideally, one would like there to exist a function $\text{Transform}_n : \mathbb{R} \to \mathbb{R}$ that

[4] For instance, a sentence `[CLS] Timi Cook is visiting Beijing now` involves three knowledge tuples 1) `Tim Cook is the CEO of Apple`; 2) `Beijing is the capital of China` and 3) `Beijing is a City`. Then a reorganized sentence in [79] is `[CLS] Tim Cook CEO Apple is visiting Beijing capital China is_a City now` with increasing position indexes: `[CLS]` in the first position, `Beijing` is the second one, and so on. By using real-valued position embeddings, we could keep the original position index for the raw sentence, and add some extra real-valued position indexes for new words: `CEO` : 2.33; `Apple`: 2.67; `capital`: 5.33; `China`: 5.67; `is_a`: 5.33; `City`: 5.67.

transforms the embedding of any word at some position pos to the embedding of a word at position $\text{pos}+n$ such that Transform_n is only dependent on the embedded value itself, but *independent* of the position pos, that is $\forall \text{pos} : g(\text{pos}+n) = \text{Transform}_n(g(\text{pos}))$.

Prior work in NLP [74], Information Retrieval [126] and Machine Learning [125] has shown the usefulness of complex numbers as richer representations. To investigate the potential of complex-valued representation, we extend the target domains of $g(\cdot)$ from $\mathbb{R}^D$ to $\mathbb{C}^D$ without losing generality, since real-valued numbers are specific complex numbers with their imaginary part being zero. This property regarding "position-free offset transformation" in complex-valued domains is formally defined in Property 4 below.

Property 4. **Position-free offset transformation**: An embedding function $g : \mathbb{N} \to \mathbb{C}$ is said to be a *position-free offset transformation* if there exists a function $\text{Transform} : \mathbb{N} \times \mathbb{C} \to \mathbb{C}$ (called the *witness*) such that for all $n \geq 1$, the function $\text{Transform}_n(\cdot) = \text{Transform}(n, \cdot)$ satisfies $\forall \text{pos} \in \mathbb{N} : g(\text{pos}+n) = \text{Transform}_n(g(\text{pos}))$. A position-free offset transformation g is said to be *linearly witnessed* if there is a function $w : \mathbb{N} \to \mathbb{C}$ such that g has a witness Transform satisfying, for all n, $\text{Transform}(n, \text{pos}) = \text{Transform}_n(\text{pos}) = w(n)$ (i.e., each Transform_n is a linear function). Or formally a translation invariance as below:

$$g(\text{pos}+n) = \text{Transform}_n(g(\text{pos})) = w(n)g(pos), \quad \forall \text{pos} \tag{4.5}$$

Additionally, a *boundedness* property is necessary to ensure that the position embedding can deal with text of any length (*pos* could be large in a long document).

Property 5. **Boundedness**: The function over the variable position should be bounded, i.e. $\exists \delta \in \mathbb{R}^+, \forall \text{pos} \in \mathbb{N}, |g(\text{pos})| \leq \delta$.

Formally, we prove the following claim that there is a unique solution that meets Properties 4 and 5 under the condition that the embedding function is linearly witnessed. We use linear functions because they are well-understood and simple with a single floating-point operation in NNs.

Claim 1 *A function $g : \mathbb{N} \to \mathbb{C}$ is a bounded and linearly witnessed position-free offset transformation iff it is on the form*

$$g(pos) = z_2 z_1^{pos} \tag{4.6}$$

for $z_1, z_2 \in \mathbb{C}$ with $|z_1| \leq 1$.

Proof: Assume that g is a bounded and linearly witnessed position-free offset transformation. Then, by linear witnessing, we have for all $\text{pos}, n_1, n_2 \in \mathbb{N}$:

$$\begin{aligned} w(n_1)w(n_2)g(\text{pos}) &= w(n_2)g(\text{pos}+n_1) = g(\text{pos}+n_1+n_2) \\ &= \text{Transform}_{n_1+n_2}(g(\text{pos})) = w(n_1+n_2)g(\text{pos}) \end{aligned}$$

whence $w(n_1+n_2) = w(n_1)w(n_2)$. Write $w(1) = z_1$ and $g(0) = z_2$. As $n_1, n_2 \in \mathbb{N}$ were arbitrary, we have $w(n) = (w(1))^n = z_1^n$ for all $n \in \mathbb{N}$. But then $g(\text{pos}+n) = w(n)g(\text{pos}) = z_1^n g(\text{pos})$. Furthermore, observe that for $\text{pos} \geq 1$, we have $g(\text{pos}) = g(1+\text{pos}-1) = w(\text{pos})g(0) = z_1^{\text{pos}} z_2 = z_2 z_1^{\text{pos}}$. For $\text{pos} = 0$, $g(0) = z_2 = z_2 z_1^0$, whence $g(\text{pos}) = z_2 z_1^{\text{pos}}$, as desired. Observe that if $|z_1| > 1$, then $g(\text{pos})$ is unbounded, whence we have $|z_1| \leq 1$. Conversely, assume that g is on the form $g(\text{pos}) = z_2 z_1^{\text{pos}}$ with $|z_1| \leq 1$. Then, $|g(\text{pos})| \leq |z_2 z_1^{\text{pos}}| \leq |z_2||z_1^{\text{pos}}| \leq |z_2|$, whence g is bounded. Define, for each $n \in \mathbb{N}$,

$w(n) = z_1^n$ and $\text{Transform}_n(\text{pos}) = w(n)\text{pos}$. Then, for all $\text{pos}, n \in \mathbb{N}$,

$$g(\text{pos} + n) = z_2 z_1^{\text{pos}+n} = z_2 z_1^{\text{pos}} z_1^n = g(\text{pos}) z_1^n = \text{Transform}_n(g(\text{pos}))$$

showing that g is a linearly witnessed position-free offset transformation. □

For any $z \in \mathbb{C}$, we may write $z = re^{i\theta} = r(\cos\theta + i\sin\theta)$. Thus, for the general form of the embedding g from Theorem 1, we have:

$$g(\text{pos}) = z_2 z_1^{\text{pos}} = r_2 e^{i\theta_2}(r_1 e^{i\theta_1})^{\text{pos}} = r_2 r_1^{\text{pos}} e^{i(\theta_2 + \theta_1 \text{pos})} \text{ subject to } |r_1| \leq 1 \tag{4.7}$$

In implementations, the above definition of g will lead to an optimization problem due to the constraint $|r_1| \leq 1$. A natural and simple way to avoid this is to fix $r_1 = 1$; note that $|e^{ix}| \equiv 1$, thus automatically satisfying the constraint, in contrast to a real-valued embedding where one would need to explicitly devise functions satisfying the constraint. Finally, Eq. 4.7 can be written in the simplified form: $g(\text{pos}) = re^{i(\omega\text{pos}+\theta)}$. Thus, one can think of g as embedding positions counterclockwise on a complex circle of radius r with a fixed period (r is the amplitude term, θ is the initial phase term, $\frac{\omega}{2\pi}$ is the frequency, and $\frac{2\pi}{\omega}$ is the period term).

4.1.3 Encoding word order in complex embeddings

We now define our complex-valued word embedding g as a map taking a word index j and position word index pos to $\mathbb{C}^D$. For a word w_j in position pos, our **general complex-valued embedding** is defined as $f(j, \text{pos}) = \boldsymbol{g}_j(\text{pos}) = \boldsymbol{r}_j e^{i(\boldsymbol{\omega_j}\text{pos}+\boldsymbol{\theta}_j)}$. Therefore, $f(j, \text{pos})$ is defined as:

$$[r_{j,1} e^{i(\omega_{j,1}\text{pos}+\theta_{j,1})}, ..., r_{j,2} e^{i(\omega_{j,2}\text{pos}+\theta_{j,2})}, \cdots, r_{j,D} e^{i(\omega_{j,D}\text{pos}+\theta_{j,D})}] \tag{4.8}$$

Note that each coordinate d ($1 \leq d \leq D$) has a separate amplitude $r_{j,d}$, period $p_{j,d} = \frac{2\pi}{\omega_{j,d}}$, and initial phase $\theta_{j,d}$. By doing so, each dimension is represented as a wave which is parameterized by an amplitude, a period/frequency, and an initial phase. The trainable parameters of the embedding are the amplitudes vector $\boldsymbol{r}_j = [r_{j,1}, ..., r_{j,D}]$, the period/frequency related weights $\boldsymbol{\omega}_j = [\omega_{j,1}, ..., \omega_{j,D}]$, and the initial phase vector $\boldsymbol{\theta}_j = [\theta_{j,1}, ..., \theta_{j,D}]$.

In our embedding, the mean vectors of $f(j, \cdot)$ taken over all positions are linearly correlated to the amplitude embedding $\boldsymbol{r}_j = [r_{j,1}, ..., r_{j,K}]$ with a coefficient $\frac{2}{\pi}$. The amplitude $r_{j,d}$ of our embedding depends only on the word w_j (and coordinate d), not on the position of the word, whence one can think of the vector $g_{pe}(j, \text{pos}) = [e^{i(\omega_{j,1}\text{pos}+\theta_{j,1})}, \cdots, e^{i(\omega_{j,D}\text{pos}+\theta_{j,D})}]$ as a "purely" positional embedding. Consequently, our complex embedding can be considered an element-wise multiplication between the word embedding $g_{we}(j) = [r_{j,1}, ..., r_{j,K}]$ and position embedding g_{pe}.

$$f(j, \text{pos}) = g_{we}(j) \odot g_{pe}(j, \text{pos}) \tag{4.9}$$

Prior works [41], [127] used mean-weight addition between word embeddings f_{we} and position embeddings f_{pe} (all words share the weights). In our work, word embeddings and position embeddings are decoupled to some extent by element-wise multiplication, and therefore the frequency/period terms (related to $\omega_{j,d}$) can adaptively adjust the importance between semantic and position information for each word and each dimension. In particular, with higher frequency (i.e., large $\omega_{j,d}$), the final embedding will change dramatically with the changing positions, while it can be fixed for any positions with an extremely-small frequency (i.e., small $\omega_{j,d}$). Interestingly, the well-known position embedding in Transformer [127]

can be seen as a degraded version of one of our specific complex word embeddings.

It is easy to know that complex-valued (position) embedding meets the boundedness property thanks to the sinusoidal activation functions. Now we check how they meet the position-free offset translation invariance.

$$\begin{aligned}\langle g_{pe}(x) \odot g_{pe}(y) \rangle &= \begin{bmatrix} e^{i\omega_1 x} \\ e^{i\omega_2 x} \\ \cdots \\ e^{i\omega_D x} \end{bmatrix} \odot \begin{bmatrix} (e^{i\omega_1 y})^\dagger \\ (e^{i\omega_2 y})^\dagger \\ \cdots \\ (e^{i\omega_D y})^\dagger \end{bmatrix} = \sum (\begin{bmatrix} e^{i\omega_1 (x-y)} \\ e^{i\omega_2 (x-y)} \\ \cdots \\ e^{i\omega_D (x-y)} \end{bmatrix}) \\ &= \sum_{d=1}^{D} e^{i\omega_d (x-y)} = \sum_{d=1}^{D} \cos(\omega_d(\mathbf{x}-\mathbf{y})) + i \sin(\omega_d(\mathbf{x}-\mathbf{y}))\end{aligned} \tag{4.10}$$

The dot product between two position vector is translation-invarant since the RHS of Eq. 4.10 only depends on the distance (see the bold term in Eq. 4.10).

Interestingly, the position-free offset translation invariance could be met by sinusoidal (position) embedding as well.

$$\langle \vec{x}, \vec{y} \rangle = \begin{bmatrix} \sin(\omega_1 x) \\ \cos(\omega_1 x) \\ \cdots \\ \sin(\omega_{\frac{D}{2}} x) \\ \cos(\omega_{\frac{D}{2}} x) \end{bmatrix} \odot \begin{bmatrix} \sin(\omega_1 y) \\ \cos(\omega_1 y) \\ \cdots \\ \sin(\omega_{\frac{D}{2}} y) \\ \cos(\omega_{\frac{D}{2}} y) \end{bmatrix} = \sum (\begin{bmatrix} \sin(\omega_1 x)\sin(\omega_1 y) \\ \cos(\omega_1 x)\cos(\omega_1 y) \\ \cdots \\ \sin(\omega_{\frac{D}{2}} x)\sin(\omega_{\frac{D}{2}} y) \\ \cos(\omega_{\frac{D}{2}} x)\cos(\omega_{\frac{D}{2}} y) \end{bmatrix}) = \sum_{i=0}^{\frac{D}{2}} \cos(\omega_i (x-y)) \tag{4.11}$$

4.1.4 Position embedding for words: rotation or translation

Since the Transformer discards the recurrent structures, position embedding is crucial to encode word order. There are two typical types of position embeddings (PEs): *translation-based PEs* and *rotation-based PEs*. Supposing that a word w_k is represented as a word vector $\mathbf{r_k}$ and its absolute position (e.g., the j-th position) is represented as $\mathbf{p_j}$. *Translation-based PEs* [127] adopt element-wise addition, i.e., $\mathbf{r_k} + \mathbf{p_j}$, with treating $\mathbf{p_j}$ as a translation shift associated by its absolute position. While *rotation-based PEs* – e.g., complex-valued word embedding introduced in [137] and Sec. 4.1.3 – adopt element-wise multiplication, i.e., $\mathbf{r_k} \odot \mathbf{p_j}$,[5] with treating $\mathbf{p_j}$ as a rotation term. Note that in both cases, the translation/rotation terms can be paremeterized by sinusoidal functions due to that its norms remain identical no matter how long the absolute position is.

Sinusoidal (position) embeddings [127] proposed a new initialization for position embedding, resulting in comparable performance with previous one [41] even without fine-tuning. The position embedding is empirically selected as

$$\begin{aligned} PE_{2k}(\cdot, pos) &= \sin(pos/10000^{2k/d_{model}}) \\ PE_{2k+1}(\cdot, pos) &= \cos(pos/10000^{2k/d_{model}}) \end{aligned} \tag{4.12}$$

where pos is the position index, $2k$ and $2k+1$ is the dimension index and d_{model} is the dimension size of embedding. The reason for choosing this position embedding was not well-explained and its general extension is unknown, leading to some difficulties to improve

[5] $\odot$ is element-wise multiplication in this paper

it.

$$WE(j) + PE(\text{pos}) = \begin{bmatrix} r_1 \\ r_2 \\ \cdots \\ r_{D-1} \\ r_D \end{bmatrix} + \begin{bmatrix} \sin \omega_1 \text{pos} \\ \cos \omega_2 \text{pos} \\ \cdots \\ \sin \omega_{D-1} \text{pos} \\ \cos \omega_D \text{pos} \end{bmatrix} = \begin{bmatrix} r_1 + \sin \omega_1 \text{pos} \\ r_2 + \cos \omega_2 \text{pos} \\ \cdots \\ r_{D-1} + \sin \omega_{D-1} \text{pos} \\ r_D + \cos \omega_D \text{pos} \end{bmatrix} \tag{4.13}$$

complex-valued (position) embeddings The proposed complex-valued (position) embeddings is defined as below:

$$WE(j) \odot PE(\text{pos}) = \begin{bmatrix} r_1 \\ r_2 \\ \cdots \\ r_D \end{bmatrix} \odot \begin{bmatrix} e^{i\omega_1 \text{pos}} \\ e^{i\omega_2 \text{pos}} \\ \cdots \\ e^{i\omega_D \text{pos}} \end{bmatrix} = \begin{bmatrix} r_1 e^{i\omega_1 \text{pos}} \\ r_2 e^{i\omega_2 \text{pos}} \\ \cdots \\ r_D e^{i\omega_D \text{pos}} \end{bmatrix} = \begin{bmatrix} r_1 \cos(i\omega_1 \text{pos}) + i r_1 \sin(i\omega_1 \text{pos}) \\ r_2 \cos(i\omega_2 \text{pos}) + i r_2 \sin(i\omega_2 \text{pos}) \\ \cdots \\ r_D \cos(i\omega_D \text{pos}) + i r_D \sin(i\omega_D \text{pos}) \end{bmatrix} \tag{4.14}$$

We claim that the proposed position embedding in [127] is a degraded version of one of our specific complex word embedding in word-sharing schema (i.e., $\omega_{j,d} = \omega_{\cdot,d}$), in which $p_{j,k} = 2\pi \times 10000^{2k/d_{model}}$ and the initial phases are set as zero. In our complex-valued position embedding, let $f_{pe,k}(\cdot, pos) = e^{i \times 10000^{2k/d_{model}}} = \cos(10000^{2k/d_{model}} pos) + i \sin(10000^{2k/d_{model}} pos)$. Note that there exists a bi-jection between $PE(\cdot, pos)$ and $f_{pe,k}(\cdot, pos)$:

$$\begin{aligned} PE_{2k}(\cdot, pos) &= \Im(f_{pe,k}(\cdot, pos)), \\ PE_{2k+1}(\cdot, pos) &= \Re(f_{pe,k}(\cdot, pos)) \end{aligned} \tag{4.15}$$

where $\Re$ and $\Im$ are the operations to take the real and imaginary part of a complex-valued number. Its inverse transformation is

$$f_{pe,k}(\cdot, pos) = PE_{2k+1}(\cdot, pos) + iPE_{2k}(\cdot, pos) \tag{4.16}$$

In our overall embedding, for each dimension $f_k(j, pos) = f_{we,k}(j) \odot f_{pe,k}(\cdot, pos)$, while it is $E_k(j, pos) = WE_k(j) + PE_k(\cdot, pos)$ in [41], [127]. Hence the main difference between position embedding in [127] and complex-valued embeddings is the fusing strategies between word components and position components: position embedding in [127] uses additions for fusion while ours use element-wise multiplication for fusion. Plus, position embeddings in [127] specifies $p_{j,k} = 2\pi \times 10000^{2k/d_{model}}$ without learning, which is a particular case of phase vectors in our complex-valued position embedding, the word-sharing schema in which all words share the same period at a certain dimension, i.e, $p_{j,k} = p_{\cdot,k}$ is irrelevant to the choice of j.

4.2 Temporal Case: Dynamic Word Embedding

Representing words in time-unrelated corpora is a common practice in the current NLP community. In some scenarios like cultural evolution, one has to consider a temporal property of word representation. In this paper, we try to move towards 'temporal word embedding'.[6] Temporal word embedding assumes that each word may have a specific meaning in a specific time, and its meaning can evolve with time. For example, the word 'gay' moves towards 'homosexual' and 'lesbian' from 1920, 'guy' moves to 'man' and 'fellow'

[6]This is also called 'Diachronic Word Embeddings' [47] or 'dynamic word embedding' [11], etc.

from 1850, 'call' moves towards 'phone' and 'message' from 1890 [47]. A recent survey of modeling lexical semantic change using computational approaches is [121].

Existing work either fails to model such dynamics [89], or align them only in two adjacent timestamps – such dynamics are separately considered only in two consequent time stamps, instead of a whole continuous process [11], [47]. Such methods considering only one-hop transformation would fit better for order-unaware embeddings, like word embeddings derived from many parallel domains (like paper collections from ACL and SIGIR) [124] – since domains are not ordered.

One word (denoted as i-th word) could have its meaning represented in a D-dimensional vector space depending on time:

$$\boldsymbol{U}_i \stackrel{def}{=} \boldsymbol{u}_i(0), \boldsymbol{u}_i(1), \boldsymbol{u}_i(2), \cdots, \boldsymbol{u}_i(t), \cdots, \boldsymbol{u}_i(T)$$

$\boldsymbol{u}_i(\cdot)$ is a mapping $\mathbb{R}_+ \mapsto \in \mathbb{R}^D$, $\boldsymbol{u}_i(t) \in \mathbb{R}^D$ is a D-dimensional vector trained on a a corpus in time t (e.g. using word2vec [89] on a newspaper collection in 1920). We define such sequence for i-th word as a matrix $\boldsymbol{U}_i \in \mathbb{R}^{D \times T}$ and all matrices form a tensor $\mathbf{U} \in \mathbb{R}^{D \times T \times V}$ while V is the size of a vocabulary (i.e. the number of words).

Existing works usually tried to connect only two consequent ones (i.e., $\mathbf{U}(t), \mathbf{U}(t+1)$) using a extra orthogonal transformations [47], a latent diffusion process [11] or a direct regularizer [148]. Note that they might lose some power when modeling such sequential evolution in the case of more than two time bins, i.e., in the case of a longer context; indeed, word meaning may evolve gradually in many years. Those methods basically assume that the transformation between $U_i(t), U_i(t+1)$ is totally independent to the one between $U_i(t+1), U_i(t+2)$. A method to unify all time series in a single dynamic process would be expected.

Each dimension (lets say j-th dimension) of $\mathbf{U}_i(t)$, denoted as $\mathbf{U}_{i,j}(t)$, is a real number (scalar). To model such word evolution in a parameter-efficient way, one has to make each dimension independent with others (i.e. $U_{i,j_1}(t)$ is independent with $U_{i,j_2}(t)$ and so on) as many distributed representation-based approaches do [89].

Therefore we could have a sequence for each dimension like

$$\mathbf{U}_{i,j} \stackrel{def}{=} \mathbf{U}_{i,j}(0), \mathbf{U}_{i,j}(1), \cdots, \mathbf{U}_{i,j}(t), \cdots, \mathbf{U}_{i,j}(T).$$

$\mathbf{U}_{i,j}(t) \in R$. Inspired by [137], word meaning could be independently learned by many functions to capture such dynamics. We propose a new paradigm to model such word meaning as functions, such that word meaning in a different time would be correlated and evolves gradually with time.

The problem becomes "how to efficiently parameterize such time vectors". The parameterization of time can be different.

More naturally, we represent the time as continuous functions such that any specific vectors will be considered as the values of such functions when the variable equals t. Examples of functions are linear or sinusoidal functions. That is, this paper aims to learn a mapping for each word w_i:

$$f : \mathbb{R} \to \mathbb{R}^D$$

$f(t)_i$ is the D-dimensional word vector for w_i in time t. In this work, we simply use linear functions and sinusoidal functions. Other functions are also worthy to be investigated. Note that the use of sinusoidal functions is motivated by their capability in terms of approximation of continuous signals; this will be discussed in detail in Sec. 4.2.3.

Time vectors	Word representation w_i	Parameter scale
global word vectors	$\boldsymbol{W}_i$	$V \times D$
Vanilla time vectors (time2vec)	$\boldsymbol{W}_i + \boldsymbol{T}_t$; $\boldsymbol{T}_t \in \mathbb{R}^D$	$V \times D + T \times D$
linear functions	$\boldsymbol{W}_i + \boldsymbol{k}t + \boldsymbol{b}$	$V \times D + 2D$
sinusoidal functions	$\boldsymbol{W}_i + \sin(\boldsymbol{\omega}t + \boldsymbol{\theta})$	$V \times D + 2D$
R&E	MLP(Trans(i) MLP(t))	$Dd^2 + 2VD + 3d^2 + Dd$
sinusoidal fun. I (Time2Fun)	$\boldsymbol{B}_i + \sin(\boldsymbol{\Omega}t)$	$VD + D$
sinusoidal fun. II (fixed freqs)	$\boldsymbol{B}_i + \boldsymbol{R}_i[\sin(\boldsymbol{\Omega}t; \cos(\boldsymbol{\Omega}t]), \boldsymbol{\Omega}_j = \frac{1}{10000}^{j/\frac{D}{2}}$	$2VD$
sinusoidal fun. III (trainable freqs)	$\boldsymbol{B}_i + \boldsymbol{R}_i[\sin(\boldsymbol{\Omega}_{\boldsymbol{i}}^{(\mathbf{1})}t); \cos(\boldsymbol{\Omega}_{\boldsymbol{i}}^{(\mathbf{2})}t)]$	$3VD$
sinusoidal fun. IV(w/ phases)	$\boldsymbol{B}_i + \boldsymbol{R}_i[\sin(\boldsymbol{\Omega}_{\boldsymbol{i}}^{(\mathbf{1})}t + \boldsymbol{\Theta}_i^{(1)}); \cos(\boldsymbol{\Omega}_{\boldsymbol{i}}^{(\mathbf{2})}t + \boldsymbol{\Theta}_i^{(2)})]$	$4VD$

Table 4.1: † denotes word-dependent time vector parameterization. One can also replace the sine functions as cosine functions.

4.2.1 Word2fun: encoding word as functions over time

The parameters in time vectors are also related to words, see Tab. 4.1.

Linear functions A simple case with linear function parameterization would be:

$$f(t)_i = \underbrace{b_i}_{\text{base term}} + \underbrace{k_i}_{\text{slope}} t$$

Sinusoidal functions Sinusoidal functions, which are commonly-used in Signal Processing field, has the ability to universally approximate many complicated functions. Inspired by [137], we propose a variant of sinusoidal functions in a time vector in t as:

$$f(t)_i = \underbrace{b_i}_{\text{base term}} + \underbrace{r_i}_{\text{amplitude}} \cos(\underbrace{\omega_i}_{\text{frequency}} t + \underbrace{\theta_i}_{\text{phase}}) \tag{4.17}$$

In practice, one can also use a combination of sine and cosine functions. Since b_i is a term not associated with time, it can be considered as static word vectors. This formulation will generally lead to additionally three times parameters than typical static word vectors. In practice, one can also ignore the amplitude term or phase, since removing the former is like a rescaling the original one,[7] and the latter was empirically evidenced to be ineffective [137].

4.2.2 Implementation in Skip-gram language model

The Skip-gram model architecture tries to predict the source context words (surrounding words) given a target word (the center word), which was considered to achieve the reverse of what the CBOW model does. Let us denote u as the target word and v is the source word, and negatively-sampled target words $\hat{\mathbb{V}} = \{\hat{v}_i\}$. It learns a dynamic word representation method f and a static word mapping g. The loss function is defined as below:

$$\mathcal{L} = -\sum_{u,v,\hat{\mathbb{V}},t} \left(\log(\delta(f(u,t)g(v)^T)) + \sum_{\hat{v}_i \in \hat{\mathbb{V}}} \log(\delta(-f(u,t)g(\hat{v}_i)^T)) \right)$$

[7] Dividing the Eq. 4.17 by r_i, we have $f(t)_i = \frac{b_i}{r_i} + \cos(\omega_i t + \theta_i)$ and then substitute $\frac{b_i}{r_i}$ as a new b_i.

The algorithm is explained in Algo. 2.

Skip-gram models basically learn two sets of word embedding, one is called "context embedding" and the other is called "target embedding". For each skip-gram (u, v), the objective is to shorten the distance between u in context embedding and v in the target embedding by maximizing their dot product. Therefore, we have $f(\cdot)$ as the context embedding and $h(\cdot)$ as the target.

The $f(\cdot)$ is used for temporal word embedding and it is therefore time-sensitive. The static compass $h(\cdot)$ is the way to align the time-specific word embedding at different times. Without the compass, time-specific word embedding may be rotated with angles – see the discussion on the Alignment Issues in Sec. 2.4.4.1 – since word embedding is arbitrarily Rotation-invariant.

One may use the context embedding f as the static one and h as the temporal one, this is also reasonable since skip-gram is a symmetric relationship between words. Due to the fact that there is no notable difference to choose which one being temporal/static, we follow [34] to make context embedding f being temporal and the target embedding h being static.

For the objective function, the loss function is a typical *cross entropy loss*. Suppose that we have a word pair u, v, its predicted probability to be a skip-gram is $\text{sigmoid}(f(u,t)h(v)^T)$; its probability not to be skip-gram is

$$1 - \text{sigmoid}(f(u,t), h(v)^T) = \text{sigmoid}(-f(u,t)h(v)^T)$$

This is because that $1-\text{sigmoid}(x) = 1-\frac{1}{1+e^{-x}} = \frac{1+e^{-x}-1}{1+e^{-x}} = \frac{e^{-x}}{1+e^{-x}} = \frac{1}{1+e^{x}} = \text{sigmoid}(-x)$. Therefore, the predicted probability distribution is

$$p_{\text{predicted}} = \begin{pmatrix} \text{sigmoid}(f(u,t), h(v)^T), \\ \text{sigmoid}(-f(u,t), h(v)^T) \end{pmatrix}$$

while its ground truth probability distribution is

$$p_{\text{pos}} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

when u, v is the correct skip-gram, or

$$p_{\text{neg}} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

when v is the negatively sampled word as a negative example.

One has to minimize the difference between the predicted probability distribution and the ground truth probability distribution. A typical way is to use the cross-entropy loss for positive examples as below:

$$\text{XEntropy}(p_{\text{predicted}}, p_{\text{pos}}) = -\log\Big(\text{sigmoid}(f(u,t), h(v)^T)\Big)$$

For negative examples,

$$\text{XEntropy}(p_{\text{predicted}}, p_{\text{neg}}) = -\log\Big(\text{sigmoid}(-f(u,t), h(v)^T)\Big)$$

Since the numbers of positive and negative examples are imbalanced: for each skip-gram pair, we have to sample $|\hat{\mathbb{V}}|$ negative samples. To balance the positive examples and

negative examples, we reweight training examples by giving a weight of 1 for the loss of positive examples and giving a weight of $\frac{1}{|\hat{\mathbb{V}}|}$ for negative examples. This results in the loss function:

$$L = -\sum_{(u,v,\hat{\mathbb{V}},t)\in\mathbb{D}_{\text{train}}} \left(\log(\delta(f(u,t)h(v)^T)) + \frac{1}{|\hat{\mathbb{V}}|} \sum_{\hat{v}_i\in\hat{\mathbb{V}}} \log(\delta(-f(u,t)h(\hat{v}_i)^T)) \right)$$

ALGORITHM 2: Training algorithm for word2fun.

Requiring timestamped corpora $\{\mathcal{C}_t\}$ with triplets (u, v, t)
Initializing a dynamic word embedding $f : (\mathbb{N}, \mathbb{R}) \to \mathbb{R}^D$
Initializing a static word embedding $g : \mathbb{N} \to \mathbb{R}^D$
shuffling timestamped corpora
while *not converge* **do**
  for *t-timestamped skip-gram* (u, v) *in* $\{\mathcal{C}_t\}$ **do**
    calculating the loss for the positive sample $\mathcal{L}_1 = \log(\delta(f(u,t)g(v)^T))$
    generating negatively-sampled target words $\hat{\mathbb{V}}$
    calculating the loss for negative samples $\mathcal{L}_2 = -\sum_{\hat{v}_i\in\hat{\mathbb{V}}} \log(\delta(-f(u,t)g(\hat{v}_i)^T))$
    back-propagating using the loss $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2$
  end
end

4.2.3 Function approximation using polynomials

For a skip-gram pair (w_i, w_j),[8] the similarity degree between a context word w_i and a target word w_j in time t is calculated as a dot product between the time-specific context embedding $f(i, t)$ and the static target embedding $h(j)$ — the latter being the 'compass'.

$$y_{i,j}(t) = f(i,t)h(j)^T \tag{4.18}$$

By treating $y_{i,j}(t)$ as a function over time, it measures the *similarity between w_i and w_j over evolving time*;[9] we argue that semantic meaning evolution can be captured by approximating arbitrary between-word similarity over time. The problem becomes how to approximate a real function over time t, i.e., $y_{i,j}(t)$, by selecting appropriated parameterization of f and h.

Let us recall Weierstrass's theorem that states that every continuous real function defined on $[a, b]$ can be approximated by a real polynomial. Note Theorem 2 also holds for trigonometric polynomial defined in Definition 1, see Corollary 1.

Theorem 2 ***Weierstrass Approximation Theorem.*** *Let F be a continuous real-valued function defined on the real interval $[a, b]$. For any $\epsilon > 0$, there exists a polynomial P such that for all $x \in [a, b]$, we have $|F(x) - P(x)|_\infty < \epsilon$.*

[8] If w_i and w_j appear together in a h-size window, we call (w_i, w_j) as a skip-gram pair in this paper, w_i is the context word and w_j is the target word, and vice versa.

[9] For PPMI factorization to obtain word vectors, $y_{i,j}(t)$ is the changing PPMI between w_i and w_j over time.

Definition 1 *A **trigonometric polynomial** of degree D is an expression of the form $\Delta + \sum_{k=1}^{D} \alpha_k \cos(kx) + \beta_k \sin(kx)$, where $\Delta, \alpha_1, \ldots, \alpha_D, \beta_1, \ldots, \beta_D \in \mathbb{R}$.*

Corollary 1 ***Approximation by trigonometric polynomials*** *For every continuous function $F : \mathbb{R} \to \mathbb{R}$ defined on the real interval $[0, 2\pi]$, and for any $\epsilon > 0$, there exists a trigonometric polynomial P such that for all $x \in [0, 2\pi]$, we have $|F(x) - P(x)|_\infty < \epsilon$.*

Corollary 2 *The trigonometric polynomials span$\{1, \sin x, \cos x, \cdots, \sin(Dx), \cos(Dx)\}$ is dense in $\mathcal{C}[0, 2\pi]$ iff span$\{1, \sin \frac{2\pi x}{a-b}, \cos \frac{2\pi x}{a-b}, \cdots, \sin \frac{2\pi Dx}{a-b}, \cos \frac{2\pi Dx}{a-b}\}$ is dense in $\mathcal{C}[a, b]$.*

From Corollary 1 [99], [150], we know that a trigonometric polynomial

$$\{0, \sin x, \cos x, \cdots, \sin(Dx), \cos(Dx)\} \tag{4.19}$$

spans a subspace that can approximate any continuous functions defined on $[0, 2\pi]$ (i.e., it is dense in $\mathcal{C}[0, 2\pi]$). By Corollary 2, one can conclude that a weighted sum of sinusoidal functions with appropriate periods can approximate any continuous functions defined in an arbitrary closed interval [99].

4.2.4 Sinusoidal Parameterization in Word2Fun

Since the static embedding $h(j)$ is not related to time t, we consider $f(i, t)$ with sinusoidal paramerization (a mixture of cosine and sine functions plus a bias term). Then Eq. 4.18 will result in [10]:

$$\begin{aligned} y_{i,j}(t) = f(i,t)h(j)^T &= \sum \left(\begin{bmatrix} \boldsymbol{B}_{i,1} + \boldsymbol{R}_{i,1} \sin(\Omega_1 t) \\ \boldsymbol{B}_{i,2} + \boldsymbol{R}_{i,2} \cos(\Omega_1 t) \\ \cdots \\ \boldsymbol{B}_{i,D-1} + \boldsymbol{R}_{i,D-1} \sin(\Omega_{\frac{D}{2}} t) \\ \boldsymbol{B}_{i,D} + \boldsymbol{R}_{i,D} \cos(\Omega_{\frac{D}{2}} t) \end{bmatrix} \odot \begin{bmatrix} \boldsymbol{C}_{j,1} \\ \boldsymbol{C}_{j,2} \\ \cdots \\ \boldsymbol{C}_{j,D-1} \\ \boldsymbol{C}_{j,D} \end{bmatrix} \right) \\ &= \underbrace{\sum_{k=1}^{D} \boldsymbol{B}_{i,k} \boldsymbol{C}_{j,k}}_{\Delta} + \sum_{k=1}^{\frac{D}{2}} \underbrace{\boldsymbol{R}_{i,2k-1} \boldsymbol{C}_{j,2k-1}}_{\alpha_{i,j,k}} \sin(\Omega_k t) + \underbrace{\boldsymbol{R}_{i,2k} \boldsymbol{C}_{j,2k}}_{\beta_{i,j,k}} \cos(\Omega_k t) \end{aligned} \tag{4.20}$$

Therefore, $y_{i,j}(t)$ is a weighted sum of sinusoidal functions plus a constant term $\Delta = \sum_{k=1}^{D} \boldsymbol{B}_{i,k} \boldsymbol{C}_{j,k}$. By replacing the coefficients with $\alpha_{i,j,k}$ and $\beta_{i,j,k}$, we can rewrite Eq. 4.20 as $y(t) = \Delta + \sum_{k=1}^{\frac{D}{2}} \alpha_{i,j,k} \sin(\Omega_k t) + \beta_{i,j,k} \cos(\Omega_k t)$; $\{\alpha_{i,j,k}\}_{k=1}^{\frac{D}{2}}$ and $\{\beta_{i,j,k}\}_{k=1}^{\frac{D}{2}}$ are the coefficients and $\{\Omega_k\}_{k=1}^{\frac{D}{2}}$ are the corresponding frequencies. Following the argument in [28] when discussing approximation properties of sine and cosine activation functions, since linear combinations of sine and cosine generate all finite trigonometric polynomials that have approximation properties described in Section 4.1, the dot product between the dynamic word embedding $f(i, t)$ and the static target embedding $h(i)$, denoted as $y_{i,j}(t)$, can capture any evolving relations between arbitrary skip-gram pairs.

By iteratively training all skip-gram pairs, Word2Fun can model complicated semantic change across time. Intuitively, small frequencies would reflect some long-range evolution, while some big frequencies would capture short-range evolution. Such periodical property would allow such functions to capture long enough evolution without considering boundedness issues. Note that the function would not necessarily be periodical with an extremely long period in a limited time span.

[10] $\odot$ is the element-wise multiplication

4.2.5 The advantages of Word2fun over the DiffTime model

The DiffTime Model [106] is not effective since the time encoder and word encoder are *decomposable*; this may lead to the two properties in contradictions with word meaning evolution that we discussed in Sec. 2.4.1.1. We first define the property of 'decomposability' for word functions.

Definition 2 ***Decomposability of word functions*** *We call a time-aware word functions* $\mathbf{U}$ *being decomposable when* $\mathbf{U}$ *can be decomposed by an arbitrary operation* δ *between a separate word encoder* f *and word encoder* g*, namely*

$$U_{i,t} = \delta(f_i, g_t) \tag{4.21}$$

Remark 1 Time2Fun is *decomposable* when δ is the addition operation. Time2Fun can be composed of a word encoder f and a time encoder g. In addition, a word i in time t is represented as

$$U_{i,t} = f_i + g_t \tag{4.22}$$

Remark 2 The DiffTime model is *decomposable* when δ is the matrix-vector multiplication operation. The DiffTime model without the last output layer can be decomposed as follows:

$$U_{i,t} = f_{word}(w) * f_{time}(t)$$

where $*$ is a matrix-vector product.

The matrix-vector product operation may be a better operation than the addition in Time2Fun as the interaction module. Moreover, the last output layer of the DiffTime model may help improve the composition of content (i.e., word) and time. This may help account for why the DiffTime model performs much better than Time2Fun, which relies on a simple interaction module, i.e., addition.

In summary, we would like to highlight that the proposed Word2fun has the following advantages over the DiffTime model.

- **Link the Weierstrass Approximation Theorem**: In Sec.4.2.3 we discussed the motivations for the use of trigonometric polynomials and their capability in terms of function approximation because of the Weierstrass Approximation Theorem.
- **Interpretability**: The learned functions provide advantages in terms of interpretability over those learned in the DiffTime model. We could use visualization to explicitly show how the learned functions look like. Note that in Word2Fun each dimension of word embedding is an individual (and also simple) function over time and functions in different dimensions are independent of each other; this simplification makes it possible to visualize learned functions dimension by dimension. The parameters for the functions like amplitude terms, frequency terms, bias terms, and initial phase terms have concrete physical meanings that could provide us an indication of the speed or of the extent to which the word meaning changes over time. Individual parameters in the DiffTime model are difficult to interpret since many linear/non-linear layers are used and it involves some complicated operations like tensor-vector product (for $Trans_w$ in Eq. 5 of the DiffTime Model paper).
- **Empirical effectiveness**: Empirical results showed that our model Word2Fun performs better than the DiffTime model in clustering, semantic analogy, and semantic shift detection tasks – see Sec. 5.3.

- **Better alignment of dynamic word embedding**: The compass that uses static embedding as target embedding could help for the alignment. The DiffTime model should suffer more from the rotation-invariant issues of word embedding since neither target embedding nor context embedding is independent of time — see Sec. 2.4.4.1.

Chapter 5

Experiments

This chapter will describe the experiments carried out to investigate the two research questions introduced in Sec. 1.2. In detail, Sec. 5.1 evaluates the Quantum Probability Driven Network and its extension CNM in text matching. Sec. 5.2 and Sec. 5.3 reports the experiments carried out to evaluate the effectiveness of the wave-like sequential modeling in the spatial scenario and the temporal scenario respectively. The experimental results reported in Sec. 5.1 were published in [134] and [74]; those reported in Sec. 5.2 were published in [137]; and those reported in Sec. 5.3 were published in [133].

5.1 Experiments for RP1: Quantum Probability-Driven Network

Sec. 5.1.1 introduces the implementation details. The experimental results in Sec. 5.1.2 and Sec. 5.1.3 shows that the proposed method could achieve comparable results with SOTA methods. Finally, Sec. 5.1.4 shows that our method has better interpretability in terms of both transparency and post-hoc explanation.

5.1.1 A QPDN implementation

In order to implement the proposed framework, we further propose an end-to-end neural network on its basis called 'Quantum Probability-Driven Network' (QPDN).

The embedding layer The parameters of embedding Layer consist of $\{R, \Phi, \Pi\}$, denoting the amplitude embedding, phase embedding and term-weight lookup table. Eq. 3.2 expresses a quantum representation as a unit-length, complex-valued vector representation for a word w, i.e. $|w\rangle = [r_1 e^{i\phi_1}, r_2 e^{i\phi_2} ... r_n e^{i\phi_n}]^T$. The term-weight lookup table is used to weight words for semantic combination, which will be described in the next subsection. During training, word embeddings need to be normalized to unit length after each batch. While it would be faster if we perform normalization after several batches [145].

The mixture layer A sentence is modeled as a density matrix, which is constructed in a bottom-up way in Sec. 3.2.3. Instead of using uniform weights in Eq. 3.3, word-specific weights are used for each word, which is commonly used in IR, e.g. inverse document frequency (IDF) as a document-dependent weight in TF-IDF scheme [119]. The new formula for the density matrix is given as $\rho = \sum_i^m p(w_i) |w_i\rangle \langle w_i|$. In order to guarantee

Table 5.1: Dataset Statistics. (CV means 10-fold cross validation for testing performance.)

Dataset	train	test	vocab.	task	Classes
CR	4K	CV	6K	product reviews	2
MPQA	11k	CV	6K	opinion polarity	2
SUBJ	10k	CV	21k	subjectivity	2
MR	11.9k	CV	20k	movie reviews	2
SST	67k	2.2k	18k	movie reviews	2
TREC	5.4k	0.5k	10k	Question	6

the unit trace length for density matrix, the word weights which are from the lookup table in a sentence are normalized to a probability value through a softmax operation:

$$p(w_i) = \frac{e^{\pi(w_i)}}{\sum_j^m e^{\pi(w_j)}} \tag{5.1}$$

Compared to IDF weight, the normalized weight for a specific word in our approach is not static, but updated adaptively in training phase. Even in the inference/test phase, the real term weight i.e. $p(w_i)$ is also not static, but highly depends on the neighbor context words through nonlinear softmax function.

The measurement layer The measurement layer adopts a set of 1-order measurement projectors $\{|v_i\rangle \langle v_i|\}_{i=1}^k$, while $|v_i\rangle \langle v_i|$ is the outer product of its corresponding state in Semantic Hilbert Space $|v_i\rangle$. After each measurement, we can obtain a measured probability for each measurement state like $q_j = tr(\rho |v_j\rangle \langle v_j|)$. Finally, we can obtain a vector $\vec{q} = [q_1, q_2, ...q_k]$. Similarly to the word vectors which are also represented as unit states, the states $|v_i\rangle$ are also normalized after several batches.

The dense layer The resulted vector $\vec{q}$ consists k probabilities (namely positive scalar numbers). $\vec{q}$ is treated as the input of the dense layer to predict the label. A dense layer with softmax activation is adopted after measurement layer to map $\vec{q}$ to label space, namely, $\widehat{\vec{y}} = \text{softmax}(\vec{q} \cdot W)$. The loss is cross-entropy loss between $\widehat{\vec{y}}$ and the one-hot ground truth label $\vec{y}$.

5.1.2 Text classification

Sec. 5.1.2.1 and Sec. 5.1.2.2 introduce the experiment setting and experiment results respectively.

5.1.2.1 Experimental Setup

Datasets Our model is evaluated on 6 datasets for text classification: CR customer review [56], MPQA opinion polarity [139], SUBJ sentence subjectivity [96], MR movie review [96], SST binary sentiment classification [117], and TREC question classification [75]. The statistics of them are shown in Tab. 5.1.

Baselines We compare the proposed QPDN with various models, including Uni-TFIDF, Word2vec, FastText [60] and Sent2Vec [95] as unsupervised representation learning baselines, CaptionRep [51] and DictRep [52] as supervised representation learning baselines, as well as

Table 5.2: Experimental Results in percentage (%). The best performed value (except for CNN/LSTM) for each dataset is in bold. where † means a significant improvement over FasText.

Model	CR	MPQA	MR	SST	SUBJ	TREC
Uni-TFIDF	79.2	82.4	73.7	-	90.3	85.0
Word2vec	79.8	**88.3**	77.7	79.7	90.9	83.6
FastText [60]	78.9	87.4	76.5	78.8	91.6	81.8
Sent2Vec [95]	79.1	87.2	76.3	80.2	91.2	85.8
CaptionRep [51]	69.3	70.8	61.9	-	77.4	72.2
DictRep [52]	78.7	87.2	76.7	-	90.7	81.0
Ours: QPDN	**81.0**†	87.0	**80.1**†	**83.9**†	**92.7**†	**88.2**†
CNN [63]	81.5	89.4	81.1	88.1	93.6	92.4
BiLSTM [26]	81.3	88.7	77.5	80.7	89.6	85.2

CNN [63] and BiLSTM [26] for advanced deep neural networks. We report the classification accuracy values of these models from the original papers.

Parameter Setting In this paper, we use Glove word vectors [97] with 50,100,200 and 300 dimensions respectively. The amplitude embedding values are initialized by L2-norm, while the phases in complex-valued embedding are randomly initialized in $-\pi$ to π. We search for the best performance in a parameter pool, which contains a learning rate in {1e-3,1e-4,1e-5,1e-6}, an L2-regularization ratio in {1e-5,1e-6.1e-7,1e-8}, a batch size in {8,16,32,64,128}, and the number of measurements in {5,10,20,50,100,200}.

Parameter Scale The main parameters in our model are amplitude embedding R and phase embedding Φ. Since both of them are $n \times |V|$ in shape, the number of parameters is roughly two times that of fastText [89]. For the other parameters, Π is $|V| \times 1$, $\{|v_i\rangle\}_{i=1}^{k}$ is $k \times 2n$, while W is $k \times |L|$ with L being the label set. Apart from word embeddings, the model is robust with limited scale at $k \times 2n + n \times |V| + k \times |L|$ for the number of parameters.

5.1.2.2 Results

The results in Tab. 5.2 demonstrate the effectiveness of our model, with improved classification accuracies over some strong baseline supervised and unsupervised representation models on most of the datasets except MPQA. In comparison with more advanced models including BiLSTM and CNN, our model generally performs better than BiLSTM with increased accuracy values on the multi-class classification dataset (TREC) and three binary text classification datasets (MR, SST & SUBJ). However, it under-performs CNN on all 6 datasets with a difference of over 2% on 3 of them (MPQA, SST & TREC), probably because that it uses fewer parameters and simpler structures. We argue that QPDN achieves a good balance between effectiveness and efficiency, due to the fact that it outperforms BiLSTM.

Ablation Test The ablation test in Tab. 5.3 aims to examine how each component influences the final performance of QPDN. All ablation settings under ablation are comparable in time cost. Particularly, a double-length real word embedding network is used

Table 5.3: Ablation Test

Setting	SST	Δ
FastText [60]	0.7880	-0.0511
FastText [60] with double-dimension real word vectors	0.7883	-0.0508
fixed amplitude part but trainable phase part	0.8199	-0.0192
replace trainable weights with fixed mean weights	0.8303	-0.0088
replace trainable weights with fixed IDF weights	0.8259	-0.0132
non-trainable projectors with fixed orthogonal ones	0.8171	-0.0220
replace projectors with dense layer	0.8221	-0.0170
QPDN	**0.8391**	-

to compare a complex-valued word embedding: 100-dimensional real-valued word vectors underperforms 50-dimensional complex-valued vectors. Mean weights and IDF weights are used as alternative word weighting strategies to check the necessity of introducing trainable weights. Experimental results shows that they slightly harm the performance. The trainable weights may better fit the data during training. Experimental results show that a set of non-trainable orthogonal projectors and a vanilla dense layer on top of the sentence density matrix also achieve good performance, although they are not as good as the trainable semantic measurements. This evidences the necessity to introduce trainable semantic measurements. In summary, Tab. 5.3 shows that each component in QPDN plays an positive important role.

Parameter sensitivity We examined the sensitivity of the QPDN model with respect to the number of measurement projectors, one of the most crucial parameters in our model. On all datasets, the accuracy values for varied numbers of measurements are shown in Tab. 5.4, with the remaining hyper-parameters being the same, i.e. 16 for batch_size, Rmsprop as the optimizer with learning rate of 0.1, 50 as the word embedding dimension and no L2 regularization. For binary classification, even a small number measurement projectors lead to comparable performance to the optimal accuracy value. In multi-label classification task (TREC), however, the performance grows steadily when the number of measurement projectors increases. We believe that it is because a measurement projector corresponds uniquely to one measurement probability while a class label is determined by a linear combination of all scalar-valued probabilities. It is therefore difficult to accurately predict multiple labels with a limited number of measurements.

Overall speaking, the model exhibits a low sensitivity to the number of measurement projectors, and therefore has a high robustness. Moreover, the measurement projectors are linearly combined to address a certain task, which makes the QPDN more transferable to other tasks, on which we plan to further investigate in the future.

5.1.3 Text matching

Sec. 5.1.3.1 and Sec. 5.1.3.2 introduce the experiment setting and experiment results respectively.

Table 5.4: Parameter Sensitivity of the number of measurement projectors

k	CR	MPQA	SUBJ	MR	SST	TREC
1	0.7646	0.7672	0.9050	0.7685	0.8309	0.3540
3	0.7804	0.8219	0.9140	0.7741	0.8221	0.6160
5	0.7725	0.8200	0.9190	0.7816	0.8298	0.5260
10	0.7725	0.8332	0.9140	0.7826	0.8320	0.8080
20	0.7804	0.8435	0.9140	0.7769	0.8292	0.8120
30	0.7884	0.8445	0.9150	0.7910	**0.8375**	0.8380
50	0.7884	0.8483	**0.9260**	0.7779	0.8314	0.8380
80	0.7910	0.8454	0.9140	0.7938	0.8265	0.8520
100	**0.8069**	0.8501	0.9230	**0.7976**	0.8281	0.8480
200	0.7963	0.8615	0.9200	0.7844	0.8270	0.8480
300	0.7963	0.8539	0.9240	0.7919	0.8287	0.8520
400	0.7910	**0.8690**	0.9230	0.7938	0.8314	**0.8620**
500	0.7910	0.8539	0.9170	0.7948	0.8276	0.8600

Table 5.5: Dataset Statistics. For each cell, the values denote the number of questions and question-answer pairs respectively.

Dataset	train	dev	test
TREC QA	1229/53417	65/117	68/1442
WikiQA	873/8627	126/130	633/2351
Yahoo QA	57403/287015	7175/35875	7175/35875

5.1.3.1 Experimental setup

Datasets The experiments were conducted on three benchmarking datasets for question answering (QA), namely TREC QA [129], WikiQA [147] and Yahoo QA[1]. TREC QA is a standard QA dataset in the Text REtrieval Conference (TREC). WikiQA is released by Microsoft Research on open domain question answering, and Yahoo QA is a community-based QA dataset. On these three datasets, we remove the questions with no correct answers. We use the same setting with [123] for Yahoo QA. In particular, for each question, we construct negative examples by sampling 4 answers from the answer pool with Okapi BM25 model. The statistics of the cleaned datasets are given in the Table 5.5. For all datasets, the task is to rank the candidate answers according to their relevance with a given question. The evaluation metrics are three rank-based metrics, namely mean average precision (MAP), mean reciprocal rank (MRR) and Precision At 1 (P@1), which are the most commonly used metrics in the respective datasets in our experiment. In particular, we use P@1 and MRR for Yahoo QA because there is only one correct answer for a question.

Baselines We conduct a comprehensive comparison across a wide range of models. On TREC QA the experimented models include Bigram-CNN [152], three-layered Long Short-term Memory (LSTM) in combination with BM25 (LSTM-3L-BM25) [138], attention-based neural matching model (aNMM) [146], Multi-perspective CNN (MP-CNN) [49], CNTN [101], attention-based LSTM+CNN model (LSTM-CNN-attn) [122] and pairwise

[1]http://webscope.sandbox.yahoo.com/catalog.php?datatype=l
&did=10

Table 5.6: Experiment Results on TREC QA Dataset. The best performed values are in bold.

Model	MAP	MRR
Bigram-CNN	0.5476	0.6437
LSTM-3L-BM25	0.7134	0.7913
LSTM-CNN-attn	0.7279	0.8322
aNMM	0.7495	0.8109
MP-CNN	0.7770	0.8360
CNTN	0.7278	0.7831
PWIM	0.7588	0.8219
QLM	0.6780	0.7260
NNQLM-I	0.6791	0.7529
NNQLM-II	0.7589	0.8254
CNM	**0.7701**	**0.8591**
Over NNQLM-II	1.48% ↑	4.08% ↑

Table 5.7: Experiment Results on Yahoo QA Dataset. The best performed values are in bold.

Model	P@1	MRR
Okapi BM-25	0.2250	0.4927
LSTM	0.4875	0.6829
CNN	0.4125	0.6323
CNTN	0.4654	0.6687
QLM	0.3950	0.6040
NNQLM-I	0.4290	0.6340
NNQLM-II	0.4660	0.6730
CNM	**0.4880**	**0.6845**
Over NNQLM-II	4.72% ↑	1.45% ↑

word interaction modeling (PWIM) [50]. On Yahoo QA dataset, we compare our CNM with the model Okapi BM25, CNN, LSTM and CNTN [101], which is reported by [123]. On WikiQA dataset, we involve the following models into comparison: Bigram-CNN [152], CNN with word count information (CNN-Cnt) [147], attention-based CNN (ABCNN) [149] and LSTM with attention (LSTM-attn) [87].

On all three datasets, we report the results of quantum language model [118] and two models NNQLM-I, NNQLM-II by [155].

Parameter settings The parameters in the network are $\Theta = \{R, \Phi, \{|v_i\rangle\}_{i=1}^k\}$, in which R and Φ denote the lookup tables for amplitudes and complex phases of each word, and $|v_i\rangle\}_{i=1}^k$ denotes the set of semantic measurements. We use 50-dimension complex word embedding.The amplitudes are initialized with 50-dimension Glove vectors [97] and L2-norm regularized during training. The phases are randomly initialized under a normal distribution of $[-\pi, \pi]$. The semantic measurements $\{|v_i\rangle\}_{i=1}^k\}$ are initialized with orthogonal real-valued one-hot vectors, and each measurement is constrained to be of unit length during training. We perform max pooling over the sentence dimension on the measurement probability matrices, resulting in a k-dim vector for both a question and

Table 5.8: Experiment Results on WikiQA Dataset.The best performed values for each dataset are in bold.

Model	MAP	MRR
Bigram-CNN	0.6190	0.6281
BILSTM	0.6557	0.6695
LSTM-attn	0.6639	**0.6828**
CNN	**0.6701**	0.6822
QLM	0.5120	0.5150
NNQLM-I	0.5462	0.5574
NNQLM-II	0.6496	0.6594
CNM	0.6548	0.6664
Over NNQLM-II	1.01% ↑	1.01% ↑

an answer. We concatenate the vectors for $l = 1, 2, 3, 4$ for a question or an answer. We will use a longer sliding window in datasets with longer sentences. The cosine similarity is used as the distance metric of measured probabilities. We use triplet hinge loss and set the margin $\alpha = 0.1$. We also use a dropout over the embedding layer and measurement probabilities with a dropout rate of 0.9.

A grid search is conducted over the parameter pools to explore the best performance. The parameter pools include the learning rates in {0.01,0.05,0.1}, batch sizes in {8,16,32}, the number of semantic measurements in {50,100,300,500}.

Parameter scale The proposed CNM is efficient with a limited scale of parameters. Apart from the complex word embeddings which are $|V| \times 2n$ by size, the only set of parameters are $\{|v_i\rangle\}_{i=1}^{k}$ which is $k \times 2n$, with $|V|, k, n$ being the vocabulary size, number of semantic measurements and the embedding dimension, respectively. In comparison, a single-layered CNN has at least $l \times k \times n$ additional parameters with l being the filter width, while a single-layered LSTM is $4 \times k \times (k + n)$ by the minimum parameter scale. To improving performance, much more complicated structures will be implemented for CNN and LSTM in practice, such as to consider varied filter length l for CNN or to stack several layers of bi-directional LSTM. Even though we also concatenate the representation of multiple window length, it does not introduce additional parameters to the network. Therefore, our network scales better than the advanced models on the CNN or LSTM basis.

5.1.3.2 Results

Tab. 5.8, 5.6 and 5.7 show the results for the three datasets, where the values in bold correspond to the best values out of all models. Our model achieves the best performances on two datasets, and performs slightly worse than the best-performed models on the remaining WikiQA Dataset, with a relative difference of 1.73% and 2.81%, respectively. This illustrates the general effectiveness of our proposed CNM.

Specifically, CNM outperforms most CNN and LSTM-based models, which have much more complicated structures and are less interpretable. The result confirms that CNM achieves a good balance between the interpretability and the performance: it works fine on both.

CNM performs better than both QLM and NNQLM on all three datasets, and becomes the state-of-the-art quantum-inspired QA model. This means that, by constructing the

Table 5.9: Ablation Test. The values in parenthesis are the performance difference between the model and CNM.

Setting	MAP	MRR
FastText-MaxPool	0.6659 (0.1042↓)	0.7152 (0.1439↓)
CNM-Real	0.7112 (0.0589↓)	0.7922 (0.0659↓)
CNM-Global-Mixture	0.6968 (0.0733↓)	0.7829 (0.0762↓)
CNM-trace-inner-product	0.6952 (0.0749↓)	0.7688 (0.0903↓)
CNM	0.7701	0.8591

network in a quantum probability driven manner, we manage to increase the interpretability of the model, reduce the parameter scale, as well as improving the effectiveness of the model. A great improvement over NNQLM-1 is observed on all three datasets, supporting our claim that trace inner product is not an effective distance metric of two density matrices.

Ablation test The ablation test aims to to examine whether each component positively contribute to the proposed CNM. *FastText-MaxPool* adopt max pooling over word-embedding, just like FastText [8]. *CNM-Real* replaces word embeddings and measurements with their real counterparts. For the real-valued models, we use embedding with double size of dimension in order to have similar amount of parameters when evaluating models. *CNM-Global-Mixture* adopts a global mixture of the whole sentence, in which a sentence is represented as a single density matrix. *CNM-trace-inner-product* replaces the trainable measurements with trace inner product like NNQLM. The test results in Tab. 5.9 demonstrate that each component plays a crucial role in the CNM model. In particular, the comparison with CNM-Real and FastText-MaxPool shows the effectiveness of introducing complex-valued components, the increase in performance over CNM-Global-Mixture reveals the superiority of local mixture, and the comparison with CNM-trace-inner-product evidences the usefulness of trainable measurements.

5.1.4 Interpretability analysis

This section discusses on interpretability in terms of transparency and post-hoc explanation. The former was only discussed with a single aspect, namely, decomposability that consider the self-explanation power for each component, as in 5.1.4.1. Additionally, some post-hoc explanation for the learned models in Sec. 5.1.4.2.

5.1.4.1 Transparency

We only discuss one aspect of transparency i.e., 'decomposability', that expect models to give concrete physical meaning for each component – we called 'self-explanation'. As is shown in Tab. 5.10, all components in our model have a clear physical meaning corresponding to quantum probability, where classical Deep Neural Network (DNN) can not well explain the role each component plays in the network. Essentially, we construct a bottom-up framework to represent each level of semantic units on a uniform Semantic Hilbert Space, from the minimum semantic unit, i.e. sememe, to the sentence representation. The framework is operationalized through superposition, mixture and semantic measurements. On the one hand, the explanation is reflected by well-designed constraints for all the components. On the other hand, some intuitive explanation can be performed on the crucial components of the network i.e, measurements, as shown in Sec. 5.1.4.2.

Table 5.10: Physical meanings and constraints

Components	DNN	QPDN
Sememe	-	basis vector / **basis state** $\{w\|w \in \mathcal{C}^n, \|\|w\|\|_2 = 1, \}$ complete &orthogonal
Word	real vector $(-\infty, \infty)$	unit complex vector / **superposition state** $\{w\|w \in \mathcal{C}^n, \|\|w\|\|_2 = 1\}$
Low-level representation	real vector $(-\infty, \infty)$	density matrix / **mixed system** $\{\rho\|\rho = \rho^*, tr(\rho) = 1\}$
Abstraction	CNN/RNN $(-\infty, \infty)$	unit complex vector / **measurement** $\{w\|w \in \mathcal{C}^n, \|\|w\|\|_2 = 1\}$
High-level representation	real vector $(-\infty, \infty)$	probabilities/ **measured probability** $(0, 1)$

5.1.4.2 Post-hoc explanation

The post-hoc explanation aims to know how a trained model works. They are threefold in this thesis. First, *Word Weighting Scheme* explicitly show how much each word contributes the overall meaning of a sentence/document. Then, the core learnable component, i.e., *Semantic Directions* could be explained by text explanation (a few words) that may be discriminative for downstream tasks. Lastly, some case study for *Matching Pattern* will intuitively show matching patterns between two text.

Word weighting scheme Tab. 5.11 shows the selected top 10 most important words and the top 10 least important ones words according to the well-trained network weights $p(w_i)$ in Eq. 5.1. The top most important words (in bold) identified by QPDN largely agree with our common sense. For example, 'mega-pixel' (CR) is an important consideration for a cell phone to many customers, 'numbing' and 'pleasuring' (SST) are clear indicators of negative and positive sentiment, while 'cost' and 'dynasty' (TREC) are likely to appear in a question with a number as the answer. On the contrary, the top 10 least important words are indeed less influential to the classification label. This, to some extent, suggests the word weighting scheme can identify discriminative words to text classification task. Empirical evaluations on the quality of the word weighting scheme are left for future work.

Tab. 5.12 shows the words selected from the top-50 most important words as well as top-50 unimportant ones. The weight of importance is based on the L2-norm of the learned amplitude embedding according to Eq. 5.1. The important words are more about specific topics or discriminative nouns, while the unimportant words include meaningless numbers or super-high frequency words. Note that some special form (e.g. plural form in the last row) of words are also identified as unimportant words, since we commonly did not stem the words.

Discriminative semantic directions In order to better understand the well-trained measurement projectors, we obtained the top 10 nearest words in complex-valued vector for each trained measurement state (like $|v_i\rangle$), using KD tree [15]. We take 5 measurements from the trained model for the MR dataset, and select words from the top 10 nearest words to each measurement. As can be seen in Tab. 5.13, the first measurement is roughly about changes over time, the second concerning being motivated or forced to do something. While the third measurement groups uncommon non-English words together. The last

Table 5.11: Selected learned important words. The upper lines refer the most important words while the bottom lines refers to the least important ones.

Dataset	Selected words
CR	**mega-pixel**, **revolution**, blackberry, turn, **horrible** speed, complaints, whiff, sale, attribute
MPQA	**ballot**, **biased**, **approval**, wants, sole storm, fleeing, efficient, taliban, strange
SUBJ	**outlandish**, **uneventful**, vows, brighter, **ultra-unrealistic** maybe, establishing, responds, manipulator, draining
MR	action/comedy, **wins**, **humorous**, **like**,europe enliven, esfera, debt, turfs, chilled
SST	**boom**, cried, overwritten, **numbing**, **pleasuring** unsettled, crowds, carey, acidity, segment
TREC	fossils, blair, **jewelry**, **cost**, **dynasty** chevrolet, priest, 1989, worlds,unanswerable

Table 5.12: Selected learned important words in TREC QA. All words are lower.

	Selected words
Important	studio, president, women, philosophy scandinavian, washingtonian, berliner, championship defiance, reporting, adjusted, jarred
Unimportant	71.2, 5.5, 4m, 296036, 3.5 may, be, all, born movements, economists, revenues, computers

two measurement also group words sharing similar meanings. It is therefore interesting to see that relevant words can somehow be grouped together into certain topics during the training process, which may be discriminative for the given task.

In the TREC QA dataset, We randomly took 5 measurements from the trained model with 10 measurements, and select words from the top 10 nearest words to each measurement vector. As can be seen in Tab. 5.14, the first three selected measurements were about position, movement verb and person name, while the following were about topic of history and rebellion respectively. To some extent, the learned measurement vectors from our proposed data-driven approach is easy to understand, compared to the LSTM cells or CNN kernels.

Matching pattern Tab. 5.15 shows how a pair of sentences match with each other under sliding windows. In a local context window, we visualize the relative weights (i.e. the weights after normalized by *softmax*) for each word with multiple degrees of darkness. The table illustrates that our model is capable of identifying true matched local windows of a sentence pair. Even the some words are replaced with similar forms (e.g. commit and committing in the last case) or meanings (e.g. change and new in the fourth case), it could be robust to get a relatively high matching score. From a empirical point of view, our model outperforms other models in situations where specific matching pattens are crucial to the sentence meaning, such as when two sentences share some unordered bag-of-word combinations. To some extent, it is robust up to replacement of words with similar ones in the Semantic Hilbert Space.

Table 5.13: The learned measurement for dataset MR. They are selected according to nearest words for a measurement vector in Semantic Hibert Space

Measurement	Selected neighborhood words
1	change, months, upscale, recently, aftermath
2	compelled, promised, conspire, convince, trusting
3	goo, vez, errol, esperanza, ana
4	ice, heal, blessedly, sustains, make
5	continue, warned, preposterousness, adding, falseness

Table 5.14: Selected learned measurements for TREC QA. They were selected according to nearest words for a measurement vector in Semantic Hilbert Space. All the words are lower.

	Selected neighborhood words for a measurement vector
1	andes, nagoya, inter-american, low-caste, kazakhstan
2	cools, injection, boiling,adrift
3	andrews, paul, manson, bair
4	historically, 19th-century, genetic, hatchback, shipbuilding
5	missile, exile, rebellion, darkness

5.2 Experiments for RP2 - Spatial Case: Encoding Word Positions

Sec. 5.2.1 and Sec. 5.2.2 introduce the experiment setting and experiment results respectively.

5.2.1 Experimental setup

Neural networks are typically given real numbers as inputs and return real numbers as outputs. To accommodate complex numbers as in- and output, we devise a complex-valued version of various neural network layers i.e. complex-valued FastText with dense layer, CNN, and RNN. Unlike existing complex-valued neural networks [125], [141], our feature layers are also converted into complex-valued layers.

Complex-valued neural networks We take FastText [60] as an example to show how to implement complex-valued neural networks by simulating complex-valued calculations with real numbers. FastText [60] is a simple and efficient neural network architecture using a dense layer over the sum of all word embeddings for general text classification. For a linear dense layer, i.e., $\boldsymbol{z} = \text{dense}(\boldsymbol{x} + i\boldsymbol{y})$, where $\boldsymbol{x} + i\boldsymbol{y}$ and $\boldsymbol{z}$ denote the complex-valued in- and output, respectively. Let $\boldsymbol{W} = \boldsymbol{A} + i\boldsymbol{B}$ and $\boldsymbol{b} = \boldsymbol{c} + i\boldsymbol{d}$ be complex-valued linear weights and bias, respectively. Then, the complex-valued dense layer is given by:

$$\boldsymbol{z} = \sigma\left(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{B}\boldsymbol{y} + \boldsymbol{c}\right) + i\sigma(\boldsymbol{B}\boldsymbol{x} + \boldsymbol{A}\boldsymbol{y} + \boldsymbol{d}) \tag{5.2}$$

Question	Correct Answer
Who is the [president or chief executive of Amtrak] ?	" Long-term success ... " said George Warrington , [Amtrak 's president and chief executive] ."
When [was Florence Nightingale born] ?	,"On May 12 , 1820 , the founder of modern nursing , [Florence Nightingale , was born] in Florence , Italy ."
When [was the IFC established] ?	[IFC was established in] 1956 as a member of the World Bank Group .
[how did women 's role change during the war]	..., the [World Wars started a new era for women 's] opportunities to ....
[Why did the Heaven 's Gate members commit suicide] ?,	This is not just a case of [members of the Heaven 's Gate cult committing suicide] to ...

Table 5.15: The matching patterns produced by CNM for specific sentence pairs in TREC QA. The darker the color, the bigger the word weight is. [and] denotes the possible border of the current sliding windows.

where σ is a real-valued activation function such as the sigmoid function. By rewriting (5.2) in matrix form [125], we obtain:

$$\begin{bmatrix} \Re(z) \\ \Im(z) \end{bmatrix} = \begin{bmatrix} \sigma(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{B}\boldsymbol{y} + \boldsymbol{c}) \\ \sigma(\boldsymbol{B}\boldsymbol{x} + \boldsymbol{A}\boldsymbol{y} + \boldsymbol{d}) \end{bmatrix} \tag{5.3}$$

where, for $z = x + iy$, $\Re(z) = x$ and $\Im(z) = y$. To save parameters and fairly compare with our real-valued baselines, the weights for real-part and imaginary-part input can be shared, i.e., $\boldsymbol{A} = \boldsymbol{B}, c = d$.

Experimental setting. We use six popular text classification datasets: CR, MPQA, SUBJ, MR, SST, and TREC (see Tab. 5.16). We use accuracy as evaluation measure based on fixed train/dev/test splits or cross validation, as per prior work. We use Fasttext [60], CNN [63], LSTM and Transformer [127] as NN baselines.[2] We use each of them:

(1) without positional information;

(2) with **Vanilla Position Embeddings (PE)**, randomly initialized and updated during training using the sum between word and position vectors [41];

(3) with **Trigonometric Position Embeddings (TPE)**, defining position embeddings as trigonometric functions as per Eq. 4.12;

(4) with **Complex-vanilla** word embeddings, where the amplitude embedding is initialized by the pre-trained word vectors, and the phrase embedding is randomly initialized in a range from $-\pi$ to π without considering word order [134];

(5) with our order-aware complex-valued word embeddings, **Complex-order**, which encode position in the phase parts, train the periods, and where the amplitude embedding is also initialized by pre-trained word vectors.

Our embedding generally has $3 \times D \times |\mathbb{W}|$ parameters with D-dimensional word vectors and $|\mathbb{W}|$ words, while previous work [89], [97] usually employs only $D \times |\mathbb{W}|$ parameters for embedding lookup tables. To increase efficiency and facilitate fair comparison with previous work, we set initial phases $\boldsymbol{\theta}_j = [\theta_{j,1}, ..., \theta_{j,D}]$ to a shared constant value (such as zero). Furthermore, the period vectors $\omega_{j,d}$ depend on word index j with length $|\mathbb{W}|$ and the coordinate index d with length D. To decrease the number of parameters, one can either

[2]Graph convolutional networks (GCNs) [13], [110] also encode positional information. We do not compare against them because they encode positional information inherently as part of the model, which makes redundant any additional encoding of positional information at the embedding level.

Dataset	train	test	vocab.	task	Classes
CR [56]	4K	CV	6K	product reviews	2
MPQA [139]	11k	CV	6K	opinion polarity	2
SUBJ [96]	10k	CV	21k	subjectivity	2
MR [96]	11.9k	CV	20k	movie reviews	2
SST [117]	67k	2.2k	18k	movie reviews	2
TREC [75]	5.4k	0.5k	10k	Question	6

Table 5.16: Dataset Statistics. CV means 10-fold cross validation. The last 2 datasets come with train/dev/test splits.

use a *word-sharing* scheme (i.e., $\omega_{j,d} = \omega_{\cdot,d}$), or a *dimension-sharing* scheme ($\omega_{j,d} = \omega_{j,\cdot}$), leading to $|\mathbb{W}| * D + |\mathbb{W}|$ and $|\mathbb{W}| * D + D$ parameters in total for the embedding layer.

We search the hyper parameters from a parameter pool, with batch size in $\{32, 64, 128\}$, learning rate in $\{0.001, 0.0001, 0.00001\}$, L2-regularization rate in $\{0, 0.001, 0.0001\}$, and number of hidden layer units in $\{120, 128\}$. We use pre-trained 300-dimensional vectors from Word2Vec [88] in all models except for Transformers. The models with trainable trigonometric position embedding produce nearly identical results compared to the non-trainable version, therefore we report the result of fixed position embeddings as per [127]. We adopt narrow convolution and max pooling in CNN, with number of filters in $\{64, 128\}$, and size of filters in $\{3, 4, 5\}$. In all Transformer models, we only use the encoder layer to extract feature information, where the layer is 1, dimension of word and inner hidden are 256 and 512 respectively, and head number is 8.

5.2.2 Results

The results are shown in Tab. 5.17. Our complex-order embeddings outperform all other variations at all times. This gain in effectiveness comes at a negligible (or non-existent) cost in efficiency. CNNs are the best performing NN as expected following [10]. Transformer NNs benefit the most from our complex-order embeddings, most likely because they are our weakest baseline. To contextualize these results, Tab. 5.18 shows the classification accuracy of five typical approaches on the same datasets (as reported in the original papers). Our complex-order embeddings outperform all methods, except for the CR dataset, where InferSent is marginally better. Overall, our approach is on a par with the SOTA in embeddings.

Ablation study We perform an ablation test (Tab. 5.19) on Transformer because it is the most common NN used with position embeddings. The two period-sharing schemas (dimension-sharing and word-sharing) slightly drop performance, because fewer parameters limit the representative power.

Adding initial phases also hurts performance, although we observed that the loss could decrease faster in early epochs compared to the setting without offset. The negative effect of initial phases may be due to periodicity, and $\boldsymbol{\omega}$ cannot be directly regularized with L2-norm penalties. The sharing schemes slightly decrease the performance with fewer parameters.

Note that the word-sharing schema outperforms the Vanilla Transformer, (both have a comparable number of parameters). If we choose $\Re(W^{Q/K/V}) = \Im(W^{Q/K/V})$, the additional parameters in the embedding layers will affect much less the whole parameter

Table 5.17: Text classification accuracy without position embeddings, with random position embeddings (PE), with trigonometric position embeddings (TPE), with complex-valued NNs without position embeddings (complex-vanilla), and with our complex-order embeddings. Superscripts §, †, ‡ and * mean a significant improvement over a baseline without position embeddings §, PE†, TPE‡ and Complex-vanilla * using Wilcoxon's signed-rank test $p<0.05$.

Method	MR	SUBJ	CR	MPQA	SST	TREC
Fasttext	0.765	0.916	0.789	0.874	0.788	0.874
Fasttext-PE	0.774	0.922	0.789	0.882	0.791	0.874
Fasttext-TPE	0.776	0.921	0.796	0.884	0.792	0.88
Fasttext-Complex-vanilla	0.773	0.918	0.79	0.867	0.803	0.872
Fasttext-Complex-order	**0.787**§†‡*	**0.929**§†‡*	**0.800**§†‡*	**0.889**§†‡*	**0.809**§†‡*	**0.892**§†‡*
LSTM	0.775	0.896	0.813	0.887	0.807	0.858
LSTM-PE	0.778	0.915	0.822	0.889	0.811	0.858
LSTM-TPE	0.776	0.912	0.814	0.888	0.813	0.865
LSTM-Complex-vanilla	0.765	0.907	0.810	0.823	0.784	0.784
LSTM-Complex-order	**0.790**§†‡*	**0.926**§†‡*	**0.828**§†‡*	**0.897**§†‡*	**0.819**§†‡*	**0.869**§†‡*
CNN	0.809	0.928	0.830	0.894	0.856	0.898
CNN-PE	0.816	0.938	0.831	0.897	0.856	0.890
CNN-TPE	0.815	0.938	0.836	0.896	0.838	0.918
CNN-Complex-vanilla	0.811	0.937	0.825	0.878	0.823	0.900
CNN-Complex-order	**0.825**§†‡*	**0.951**§†‡*	**0.852**§†‡*	**0.906**§†‡*	**0.864**§†‡*	**0.939**§†‡*
Transformer w/o position embedding	0.669	0.847	0.735	0.716	0.736	0.802
Transformer-PE	0.737	0.859	0.751	0.722	0.753	0.820
Transformer-TPE [127]	0.731	0.863	0.762	0.723	0.761	0.834
Transformer-Complex-vanilla	0.715	0.848	0.753	0.786	0.742	0.856
Transformer-Complex-order	**0.746**§†‡*	**0.895**§†‡*	**0.806**§†‡*	**0.863**§†‡*	**0.813**§†‡*	**0.896**§†‡*

scale in the multiple-layer Transformer, since an embedding layer is only used in the first layer instead of the following Transformer layers.

5.3 Experiments for RP2 - Temporal Case: Dynamic Word Embedding

Sec. 5.3.1 introduces the experiment setup. Sec. 5.3.2 and 5.3.3 introduce the quantitative and qualitative results of the proposed Word2Fun. Sec. 5.3.4 discusses the interpretability of Word2Fun.

5.3.1 Experimental setup

Corpora. The diachronic corpora used in this paper are reported in Tab. 5.20. The Corpus of Historical American English (**COHA**) [30] is the largest structured corpus of historical English (the 1820s-2010s), contains more than 475 million words, and is balanced by genre decade by decade. The **New York Times** (NYT) [148] contains 99,872 articles published between January 1990 and July 2016; besides the article text, metadata including title, author, release date, and section label were also collected.

Baselines. We choose the baselines from [148]:

Table 5.18: Text classification accuracy. ⋆ means that scores are reported from other papers.

Method	MR	SUBJ	CR	MPQA	SST	TREC
Word2vec Bow [26] ⋆	0.777	0.909	0.798	0.883	0.797	0.836
Sent2Vec [95] ⋆	0.763	0.912	0.791	0.872	0.802	0.858
QuickThoughts [80] ⋆	0.824	0.948	0.860	0.902	-	0.928
InferSent [26] ⋆	0.811	0.924	**0.863**	0.902	0.846	0.882
QPDN [134] ⋆	0.801	0.927	0.810	0.870	0.839	0.882

Table 5.19: Ablation test for Transformer, showing the effect of (i) the definition of embedding layer($f_d(j, \text{pos})$), and (ii) whether the real-part and imaginary transition share the weights, i.e., $\Re(W^{Q/K/V}) = \Im(W^{Q/K/V})$.

Method	Setting		Params	Accuracy	Δ
	$f_d(j, \text{pos})$	share in $W^{Q/K/V}$			
Transformer-complex-order	$r_{j,d}e^{i(\omega_{j,d}\text{pos})}$	×	8.33M	**0.813**	-
adding initial phases	$r_{j,d}e^{i(\omega_{j,d}\text{pos}+\theta_{j,d})}$	×	11.89M	0.785	-0.028
dimension-sharing period schema	$r_{j,d}e^{i\omega_{j,\cdot}\text{pos}}$	×	5.82M	0.797	-0.016
word-sharing period schema	$r_{j,d}e^{i\omega_{\cdot,d}\text{pos}}$	×	5.81M	0.805	-0.008
dimension-sharing amplitude schema	$r_{j,\cdot}e^{i\omega_{j,d}\text{pos}}$	×	5.82M	0.798	-0.015
word-sharing amplitude schema	$r_{\cdot,d}e^{i\omega_{j,d}\text{pos}}$	×	5.81M	0.804	-0.009
w/t encoding positions (complex-vanilla)	$r_{j,d}e^{i\omega_{j,d}}$	×	9.38M	0.764	-0.049
dimension-sharing period schema	$r_{j,d}e^{i\omega_{j,\cdot}\text{pos}}$	✓	4.77M	0.794	-0.019
word-sharing period schema	$r_{j,d}e^{i\omega_{\cdot,d}\text{pos}}$	✓	4.76M	0.797	-0.016
dimension-sharing amplitude schema	$r_{j,\cdot}e^{i\omega_{j,d}\text{pos}}$	✓	4.77M	0.792	-0.021
word-sharing amplitude schema	$r_{\cdot,d}e^{i\omega_{j,d}\text{pos}}$	✓	4.76M	0.801	-0.012
w/t encoding positions (complex-vanilla)	$r_{j,d}e^{i\omega_{j,d}}$	✓	8.33M	0.743	-0.07
vanilla Transformer [127]	$WE_{j,d} + PE_d$	-	4.1M	0.761	-0.052

- *Static Word2Vec*: the standard word2vec embeddings [89], trained on the entire corpus and ignoring time information.
- *Transformed Word2Vec* [66]: the embeddings are first trained separately by factorizing PPMI matrix for each year, and then transformed by optimizing a linear transformation matrix which minimizes the distance between two consequent time-stamped trained embeddings for the k nearest words' embeddings to the querying words.
- *Aligned Word2Vec* [47] : the embeddings are first trained by factorizing the PPMI matrix for each year t, and then aligned by searching for the best orthonormal transformation between two consequent time-stamped trained embeddings.

Table 5.20: Statistics of Diachronic corpora.

corpus	num. of tokens	time range	time granularity	T	Vocab
COHA [30]	472M	1810 - 2009	every decade	20	43, 734
New York Times [148]	105M	1990 - 2016	yearly	27	20, 314

- *Dynamic word embedding* [148] learns word embedding by factorizing the PPMI matrix in individual time; plus, it imposes a regularizer to encourage two subsequent word embedding being similar for alignment. We only implement dynamic word embeddings in [148], since it is a generalized version of many train-and-align paradigm-based dynamic word embeddings, e.g., [47], [66].

In addition to the baselines in [148], we considered also

- *Compass aligned word embedding* [34] that proposes a identical fixed target embeddings to align time-specific context embeddings.
- DiffTime [106] treat time as a continuous variable and train a multiple-layer neural network model for time-specific word representation.

Experimental settings. We removed words that appear less than 200 times, as [148] did. We used the Adam optimizer with a learning rate of 0.0025. The batch size can be as big as it achieves the upper bound of memory of GPU. The model converged in less than 20 epochs with no drop in terms of loss. We computed the average performance on the last three saved checkpoints. For sinusoidal functions, we used a mixture of cosine functions and sine functions. The dimension was 100; we preliminary found that models with bigger dimensions (e.g. 200 or 300) can slightly improve performance. Word2Fun with all settings was slightly slower than the standard Word2Vec due to additional computation for obtaining word vectors. However, all methods include Word2Vec and Word2Fun took 10-15 minutes for one epoch in a single Nvidia V100 32G GPU for NYT dataset.

5.3.2 Quantitative evaluations

The quantitative evaluation is performed using the quantitative methods by [148] – time-aware word clustering and temporal analogy – and the Semantic Change Detection task [111].

5.3.2.1 Time-aware word clustering

If one word is extremely frequent in a particular section, it was labeled as most-used in that section.[3] Such section annotation was used to evaluate the clustering results, which were firstly reported by [148]. Across 11 sections, there were 1,888 triplets denoted as (w_i, t_i, s_i) to indicate that word w_i in year t_i was associated with section s_i.

We apply spherical K-means on learned time-specific word vectors using cosine distance, with $K = 10$, 15, and 20 clusters. As in [148], we used *Normalized Mutual Information* (NMI) and F_β*-measure* to measure the consistency between section label and clustering results. **NMI** was defined as $\text{NMI}(\mathcal{C}, \mathcal{S}) = 2\text{MI}(\mathcal{C}, \mathcal{S})/(\text{E}(\mathcal{C}) + \text{E}(\mathcal{S}))$, where $\mathcal{S}$ denotes section labels ($\{s_i\}$) for word-year pairs $\{(w_i, t_i)\}$ and $\mathcal{C}$ denotes clustering categories ($\{c_i\}$) for these word-year pairs $\{(w_i, t_i)\}$. $\text{MI}(\cdot, \cdot)$ and $\text{E}(\cdot)$ are the functions to compute Mutual Information and Entropy respectively. $F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P+R}$ measures the effectiveness as a β-weighted harmonic mean of precision and recall. As in [148], we set $\beta = 5$ to emphasise recall to penalizing false negative more.

[3] News articles in NYT dataset are tagged with their 'sections' such as 'Business', 'Sports', 'World', and 'Technology'. For example, we see that *amazon* occurs 41% of the time in *World* in 1995, associating strongly with the rainforest, and 50% of the time in *Technology* in 2012, associating strongly with *e-commerce*.

Table 5.21: Experimental results of Time-aware word clustering.

Method	10 Clusters		15 Clusters		20 Clusters	
	NMI	F_β	NMI	F_β	NMI	F_β
Global/static word vector [89]	0.6736	0.6163	0.6867	0.7147	0.6713	0.7214
Transformed Word2Vec [66]	0.5175	0.4584	0.5221	0.5072	0.5130	0.5373
Aligned Word2Vec [47]	0.6580	0.6530	0.6618	0.7115	0.6386	0.7187
Dynamic Word2Vec [148]	0.7175	0.6949	0.7162	0.7515	0.6906	0.7585
Compass aligned Word2Vec [34]	0.5191	0.3750	0.5062	0.4051	0.5077	0.4331
DiffTime [106]	0.5726	0.5530	0.5947	0.6571	0.5877	0.6883
Word2Fun linear	0.1676	0.1813	0.2826	0.3035	0.2473	0.2932
Word2Fun I (Time2Fun)	0.1703	0.1783	0.2691	0.2680	0.2842	0.2649
Word2Fun II	**0.7281**	**0.7147**	**0.7181**	0.7645	**0.7012**	0.7616
Word2Fun III	0.7233	0.7080	0.7086	**0.7701**	0.6980	**0.7630**
Word2Fun IV	0.7111	0.6913	0.7023	0.7451	0.6823	0.7602

Table 5.22: Experimental results of temporal analogy in *test1*

Method	MRR	P@1	P@3	P@5	P@10
Global/static Word2Vec [89]	0.3560	0.2664	0.4210	0.4774	0.5612
Transformed Word2Vec [66]	0.0920	0.0500	0.1168	0.1482	0.1910
Aligned Word2Vec [47]	0.1582	0.1066	0.1814	0.2241	0.2953
Dynamic Word2Vec [148]	0.4222	0.3306	0.4854	0.5488	0.6191
Compass aligned Word2Vec [34]	**0.481**	**0.404**	**0.534**	0.582	0.636
DiffTime [106]	0.3357	0.2638	0.3493	0.4071	0.4896
Word2Fun linear	0.3016	0.2649	0.3255	0.3426	0.3630
Word2Fun I (Time2Fun)	0.3735	0.2646	0.4300	0.4955	0.5874
Word2Fun II	0.4061	0.2756	0.4916	0.5614	0.6434
Word2Fun III	0.4354	0.3076	0.5330	**0.5837**	**0.6647**
Word2Fun IV	0.4208	0.2954	0.5076	0.5715	0.6470

Experimental results. As shown in Tab. 5.21, the proposed Word2fun II, III and IV outperformed all the baseline methods. Word2Fun with linear parameterization and Word2fun I, and the DiffTime model did not perform well, since they could not learn word-dependent evolution. Word2fun IV achieved better performance than III, thus showing that adding phase was not beneficial to Word2Fun.

5.3.2.2 Temporal analogy

Temporal Analogy [120] is task which utilizes quadruples $(w^{(1)}, t^{(1)}, w^{(2)}, t^{(2)})$ to say that "$w^{(1)}$ in year $t^{(1)}$" is like "$w^{(2)}$ in year $t^{(2)}$". To examine the quality of temporal analogy, [148] created a task to investigate equivalences across years. For example, given *obama* in *2012*, we aim to find its equivalent word in 2002. As we know *obama* was the U.S. president in 2012, its equivalent word in 2002 is *bush*, who was the U.S. president in 2002. In this way, there are two test sets. The first set (test1) is based on publicly recorded knowledge that lists different names for a particular role, such as U.S. president for each

Table 5.23: Experimental results of temporal analogy in *test2*

Method	MRR	P@1	P@3	P@5	P@10
Global/static Word2Vec [89]	0.0472	0.0000	0.0787	0.0787	0.2022
Transformed Word2Vec [66]	0.0664	0.0404	0.0764	0.0989	0.1438
Aligned Word2Vec [47]	0.0500	0.0225	0.0517	0.0787	0.1416
Dynamic Word2Vec [148]	0.1444	0.0764	0.1596	0.2202	0.3820
Compass Aligned Word Embedding [34]	0.1361	0.0749	0.1918	0.2904	0.3918
DiffTime [106]	0.0868	0.0000	0.1014	0.1425	0.2548
Word2Fun linear	0.0425	0.0137	0.0384	0.0630	0.1014
Word2Fun I (Time2Fun)	0.0992	0.0000	0.1315	0.1726	0.2849
Word2Fun II	0.1194	0.0358	0.1075	0.2219	0.3863
Word2Fun III	**0.1824**	**0.0795**	**0.1973**	**0.2932**	**0.4164**
Word2Fun IV	0.1536	0.0548	0.1562	0.2411	0.3918

year. Human experts generated the second test (test2) to explore emerging technologies, brands, and significant events (e.g., disease outbreaks and financial crisis) , etc. Reciprocal Rank (MRR) and Precision@K (P@K) are used for the evaluation.

Experimental results. As shown in Tab. 5.22 and Tab. 5.23,Word2Fun III outperformed all baselines in *test2* and some metrics (including P@5, P@10) in *test1*. In *test1*, Word2Fun performed worse than Compass-aligned Word2Vec [34] in terms of MRR and P@1; this might be explained by 1) Word2Fun has fewer parameters than [34].

5.3.2.3 Semantic change detection

Effectiveness in Semantic Change Detection was investigated by the protocol of the Semeval-2020 task titled 'Unsupervised lexical semantic change detection' [111] that annotates the semantically-shift degree of some English words from 1810-1860 timespan to another timespan 1960-2010. We split data of every decade into time bins. We used Word2Fun trained on COHA dataset. For each word, we take the cosine distance between its first five decade-specific word vectors and the last five decade-specific word vectors as the semantic change degree. Thirty-seven English words were annotated with semantic shift degree by human experts. In this paper, we use the Pearson and Spearman correlation.

Semantic change evaluation is indeed not a standard setup. However, the standard setting does not fit the selected baselines [6,9,14,16,26] and the proposed model. In detail, the standard setting only provides two groups of text: one is in the range of 1810–1860, the other is in the range of 1960–2010. There is no time-specific text between the two time spans, namely 1860-1960; this hinders baselines from alignments. Indeed, the alignments of the baselines including [47], [66], [148] need an anchor to force two consecutive time-specific embeddings to be close. However, the aligned anchor is missing due to the time gap/jump between 1860-1960. Also, the original corpora in SemEval do not provide the exact timestamp for all documents. This means we could have only two data points to learn functions. To be rigorous, we also train our models in the standard corpora provided by SemEval. Word2Fun models (in Word2Fun III setting) do not converge at all, and they achieved nearly zero correlations.

Table 5.24: Semantic change detection. Baselines in the first group are implemented by this work.

models	Pearson	Spearman
Global/static Word2Vec [89]	nan	nan
Transformed Word2Vec [66]	0.0727	0.0865
Aligned Word2Vec [47]	0.3333	0.3083
Dynamic Word2Vec [148]	0.2727	0.2877
Compass aligned word embedding [34]	0.3199	0.2567
DiffTime [106]	0.1724	0.1682
Word2Fun linear	-0.1200	-0.0790
Word2Fun I (Time2Fun)	0.3925	0.4550
Word2Fun II	0.4478	**0.5038**
Word2Fun III	**0.5355**	0.4057
Word2Fun IV	0.4483	0.3578
multilingual BERT [107] (SemEval-2020 1st)	-	0.436
ensemble between aligned Word2Vec and BERT [100] (SemEval-2020 2nd)	-	0.422

Experimental Results. As shown in Tab. 5.24, Word2Fun outperformed the five mentioned baselines and the most effective approaches in the Semeval-2020 'Unsupervised lexical semantic change detection' task. Note that the Semeval-2020 participants [100], [107] did not train on the same text collection; they used a subset of COHA and we use a complete one.

5.3.2.4 More discussions on the DiffTime model

Gradient explosion issue in the DiffTime model We also noticed that the training of DiffTime is not stable, since it may suffer from gradient explosion, especially when the batch size is small. We also observed that gradient explosion can be reduced by using bigger batch sizes and gradient clips. We suspect that the tensor-vector and matrix-vector product operations may account for gradient explosion since the results of the operations of DiffTime

$$Trans_w = T\vec{w} + B$$

$$h_3 = Trans_w(timevec(t))$$

$$use_W(w,t) = M_4 h_3 + b_4$$

are not well-bounded after tensor-vector and matrix-vector transformation (see the RHS of the above equations). In contrast, sinusoidal functions are bounded in Word2Fun thanks to its periodical properties; dot product in Word2Fun is also bounded thanks to the sigmoid activation.

Comparison for parameter scale between Word2fun and DiffTime Let us define D as the dimension of word embedding (i.e, 300) and d as the hidden dimension (i.e, 100) in DiffTime. Therefore, the DiffTime model has $\mathcal{O}(Dd^2 + 2VD + 3d^2 + Dd)$ parameters. Word2Fun has $\mathcal{O}(kVD)$ parameters. Since V is 20k or 40k in the experimental corpora and k is 2-5, we could roughly conclude that the number of parameters of the DiffTime model and that of Word2Fun III is comparable. Time2Fun has fewer parameters.

5.3.3 Qualitative analysis

In Tab. 5.25, we report some results from the trained word representation in the NYT dataset that ranges from 1990 to 2009. From the the case of '*apple*', we can see that '*apple*' were more about a fruit used to prepare *sorbet* or *chutney*. Since the invention of Android in the early 2000s, '*apple*' has become more related to *Android*. The same trend can be seen in the case of 'browser': it was used in personal computers (e.g., *netscape* browser) in the PC era, and then became more popular in mobile devices (e.g., in *Android* system) in the mobile Internet era. As for '*phone*', it typically referred to a traditional *telephone* in 1990s, and it has become related to *911* (an universal emergency number) since 9/11 attacks. In the mobile internet era, *phone* is highly related to password since intelligent phones need password while the fixed-line *phone* does not need. These cases suggest that the proposed Word2Fun can model such semantic change to some extent.

Table 5.25: Most similar words for *apple*, *european*, *phone*, and *browser* from the year 1990 (denoted as 90) to 2016 (denoted as 16). All words are lower cased.

year	90	91	92	93	94	95	96	97	98	99	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
apple	sorbet										chutney										android						
browser	netscape															android											
phone	telephone										911										password						
european	republics				euro			republics			euro			hollande		republics					euro						

A detail example for a multi-sense word in semantic shift Tab. 5.26 reports on a case study to show that the proposed Word2Fun could at least to some extent capture word meaning change regarding words with multiple senses. We used the word `gay` for the case study.

word	1900s	1920s	1940s	1960s	1980s	2000s
frolicsome	**0.5230**	0.3574	0.2802	0.1511	0.1649	0.1992
playful	0.4094	0.3757	**0.4268**	0.3298	0.2425	0.2839
debonair	0.3840	0.4705	**0.5523**	0.4597	0.2243	0.3547
activists	0.2319	0.2430	0.0892	0.2894	**0.4698**	0.4072
homosexuality	-0.1435	-0.0274	0.1209	0.2605	0.3242	**0.3727**

Table 5.26: The similarity to the word `gay` over time.

Table 5.26 reports words similar to `gay` by using the representation obtained through Word2Fun III and the following measure of similarity:

$$sim(\vec{w}_1, \vec{w}_2) = \frac{\vec{w}_1 \vec{w}_2^T}{|\vec{w}_1| \cdot |\vec{w}_2|}$$

From Table 5.26, we could see that the sense of `playful` decreases over time, while the sense of `homosexuality` increases. This shows the proposed Word2Fun has some potential to deal with the evolution of sense shift.

5.3.4 Interpretability of the learned functions

We report some results we obtained from the analysis of the degree of semantic shift that can be investigated using the human annotations provided by SemEval 2020 Unsupervised Lexical Semantic Change Detection Subtask 2. The basic rationale for using the degree of change is the following: if in a given time interval (100 years in our case) the degree of change is higher, then the change speed is going to be higher; basically, we computed the "average speed" as the ratio of the total distance covered (the degree of change) and the total time taken (the time interval in our corpus). Meaning change speed of semantically-shifted words should be faster than semantically non-shifted words since the speed of the latter should be negligible.

This is quantitatively evaluated by correlations with human annotations from SemEval 2020 Unsupervised Lexical Semantic Change Detection Subtask 2.

Learned parameters	Definition	Pearson	Spearman
frequencies term	AVG($\lvert\mathbf{\Omega_i}\rvert$)	0.0958	0.1748
amplitude term	AVG($\lvert\mathbf{R_i}\rvert$)	0.1213	0.0474
bias term	AVG($\lvert\mathbf{B_i}\rvert$)	-0.3640	-0.2636
amplitude/bias terms	AVG($\lvert\mathbf{R_i}\rvert$)/AVG($\lvert\mathbf{B_i}\rvert$)	0.4141	0.2358

Table 5.27: The parameters could directly reflect the semantic shift degree.

To show how these learned parameters reflect the meaning change speed, we take the best-performed Word2Fun as an example, namely Word2Fun III, a word w_i is represented as $\mathbf{B_i} + \mathbf{R_i}[\sin(\mathbf{\Omega_i^{(1)}}t); \cos(\mathbf{\Omega_i^{(2)}}t)]$ where $\mathbf{B_i}, \mathbf{R_i}, \mathbf{\Omega_i} \in \mathbb{R}^D$. There are three cases that such a sinusoidal function degrade to a constant function: a) $\mathbf{B_i} = \infty$; b) $\mathbf{R_i} = \mathbf{0}$; or c) $\mathbf{\Omega_i} = \mathbf{0}$. Intuitively, smaller $\mathbf{B_i}$, bigger $\mathbf{R_i}$ or $\mathbf{\Omega_i}$ indicate that the word w_i is sensitive to time. In other words, *the meaning of words more likely changes over time in the case of smaller* $\mathbf{B_i}$, *bigger* $\mathbf{R_i}$ *or* $\mathbf{\Omega_i}$.

To examine the above intuition, we define the average of the absolute values for these parameters (i.e., AVG($\mathbf{B_i}$), AVG($\mathbf{R_i}$), and AVG($\mathbf{\Omega_i}$), the average AVG is computed over all the dimensions of vectors) as the indicator of semantically-shift degree of a word w_i. Since in Word2Fun III, $\mathbf{B_i}$ is the time-unrelated term while $\mathbf{R_i}[\sin(\mathbf{\Omega_i^{(1)}}t); \cos(\mathbf{\Omega_i^{(2)}}t)]$ is the time related term. We also take the ratio between AVG($\lvert\mathbf{R_i}\rvert$) and AVG($\lvert\mathbf{B_i}\rvert$) as an indicator (in the last row in Table 5.27) Table 5.27 shows that the empirical evaluation confirms our intuition. Especially, the Pearson correlations of the last two rows, especially AVG($\lvert\mathbf{R_i}\rvert$)/AVG($\lvert\mathbf{B_i}\rvert$), are significant with $p < 0.05$. However, the result is not as good as the results in Table 6. The result in Table 5.27 is surprising since these indicators do not consider the two specific evaluation time spans and they, therefore, capture the word evolution speed over the whole time span. Visualization of learned functions will be reported in Appendix B.

We also try to use the first-order derivative of learned functions to measure the word meaning change speed. The first-order derivative for the dimensions with sine functions is calculated as

$$f'(u,t) = \mathbf{R_i}\mathbf{\Omega_i^{(1)}} \cos(\mathbf{\Omega_i^{(1)}}t)$$

and The first-order derivative for the dimensions with cosine functions is calculated as

$$f'(u,t) = -\mathbf{R_i}\mathbf{\Omega_i^{(2)}} \sin(\mathbf{\Omega_i^{(2)}}t)$$

The final indicator is designed as an average of the first-order derivative in a time span $[a, b]$, as Rosenfeld & Erk did. We use p-norm to get positive numbers

$$\text{derivative}(a, b) = \text{AVG}(\sum_{t=a}^{t=b} |f'(u, t)|_p)$$

We only consider the discrete timestamp like in (1810s, 1820s, ..., 2000s). The result is reported as below:

derivative	p	timespan (a,b)	Pearson	Spearman
derivative	1	1810s - 2010s	0.2175	0.1174
derivative	1	1860s - 1960s	0.1978	0.1043
derivative	2	1810s - 2010s	0.1293	0.0908
derivative	2	1860s - 1960s	0.1318	0.1104

Table 5.28: First-order derivative of learned functions to measure semantic shift degree.

The performance in Tab. 5.28 is not as good as Tab. 5.27. We suspect that this is because the bias terms $\mathbf{B_i}$ were not considered in the first-order derivative, while the bias terms are highly related to the semantic-shifted degrees.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we proposed a quantum mechanical framework to unify semantic units in different granularity (i.e., semantic bases, lexical representation, semantic composition and semantic abstraction) in a single semantic space, thanks to the analogy between words and particles using quantum probability. The framework is implemented by a 'quantum probability driven network' (called QPDN) and by its extension for text matching (called CNM), where each of their component could have a concrete probability-grounded meaning.

To investigate the dynamic aspect of words, we formalized it as a general problem to *encoding sequential information in vector space*. This thesis discusses two cases for the general problem: a spacial case involving position embeddings and a temporal sequence involving dynamic word embedding). Interestingly, QPDN and CNM naturally induce complex-valued components due to the quantum probability theory. Such complex-valued components (with amplitude terms and phase terms) could model sequence (both for spacial sequence and temporal sequence) by directly encoding sequential order in phase terms, since the rotation nature of phases in waves makes sequential encoding always bounded.

The methods proposed in this thesis not only improve interpretability but also improve effectiveness in text classification, text matching and dynamic word meaning modeling that could support expert users. Especially, Word2Fun as dynamic word embedding is promising in a variety of tasks (e.g., clustering, temporal analogy, semantic change detection, etc.) that can support expert users (e.g., the case study for 'gay' as shown in Table 5.26).

As side effects, such wave-like modeling for spatial and temporal sequences could reinterpret commonly-used yet 'magic' sinusoidal position embedding in a principled way. Second, it motivates us to model words as sinusoidal signals especially in the scenario of dynamic word embedding.

When encoding sequential order in phase terms of complex-valued word embedding, the difference between real-valued embedding and complex-valued embedding can be roughly considered as wave-particle duality for micro objects as particles and waves. From a static point of view, the word in a specific time or position may be a fixed point (denoted as a vector) in a vector space; when we check it in a longer time span or a bigger spacial span, it could be modeled as an evolving wave and its concrete states depends on when or where you measure it. The rationale of the thesis implicitly convey a hypothesis as below:

> **Wave-particle duality**: Inspired by the wave-particle duality, words can be analogously modeled (1) as static **particles** for probabilistically-grounded

interpretation, and (2) as sequential **waves** to explicitly capture their spatial and temporal context.

The implementations of the this thesis are all open-sourced in Github that allows others reproduce results of this thesis.

- The implementations of QPDN and CNM for Sec. 5.1 are publicly available in `https://github.com/wabyking/qnn`
- The implementation of Complex-valued word embedding to encode word orders for Sec. 5.2 is publicly available in `https://github.com/iclr-complex-order/complex-order`.
- The implementation of Word2Fun for Sec. 5.3 is publicly available in `https://github.com/wabyking/word2fun`.

6.2 Future work

As future work, the thesis encourages the following directions:

Extending wave-like sequential model to general time-series modeling There are various applications of a wave-like sequential model like Word2fun. One of the most interesting applications with dynamic user profiles. In a typical static item recommendation scenario, the basic goal is to approximate user-item preference scores by a dot product between a static user embedding (denoted as $\vec{u}$) and item embedding (denoted as $\vec{i}$). $f_{u,i} \propto \vec{u}\vec{i}^T$, while $p_{u,i}$ is a scalar that indicates rating score or buying/like behaviors. When extending it to temporal process, the temporal preference $p_{u,i}(t)$ would be sequences of scalars that indicates user-item interaction behavior over time; such that the user profiles would also evolve with time t, denoted as $\vec{u}(t) = f(t)$. This could be an equivalent problem as stated in this paper and $p_{u,i}(t)$ could be approximated by a sum of sinusoids. More interestingly, the periodical properties of sinusoidal functions could be used to model repeating purchase behavior, for example, one may buy beers every month and buy coats every winter.

Processing words in wave-based computing devices Language, as a sequence of textural tokens, may be modeled as wave-like signals with amplitudes and phases. This may make use of existing wave-based computing devices for both effectiveness and efficiency purposes. Wave-based computing devices are widely seen (e.g., [36]) and attract attention in recent years. Recently [154] explores an Optical Neural Chip (ONC) that implements truly complex-valued neural networks. This thesis may provide a possibility to embedded words as complex-valued features that could be directly used in ONC.

QNLP Quantum computing may open a new door to enlarge NLP models (like pre-trained language models) since quantum computing could provide some potential for efficiency. Very recently, researchers from Oxford University and Cambridge Quantum Computing have proposed some prototype NLP models running in quantum computers [24], [81]. This shows the application of Quantum computing in NLP is not that far and encourages more exploration in this area.

Linguistic phenomenon modeled as quantum process Before the quantum age using quantum computers, some preliminary exploration of quantum probability in classical computers is beneficial. Especially in NLP, some linguistic phenomena could find some analogy with the quantum phenomenon (includes but is not limited to entanglement and interference) [20], [126], [136]. This may help us to better understand language.

Bibliography

[1] D. Aerts and S. Sozzo, "Quantum Entanglement in Concept Combinations," en, *International Journal of Theoretical Physics*, vol. 53, no. 10, pp. 3587–3603, Oct. 2014, ISSN: 0020-7748, 1572-9575. DOI: 10.1007/s10773-013-1946-z.

[2] J. Aitchison, *Language change: Progress or decay?* Cambridge university press, 2001.

[3] C. Allen, I. Balazevic, and T. Hospedales, "What the vec? towards probabilistically grounded embeddings," *Advances in Neural Information Processing Systems*, vol. 32, pp. 7467–7477, 2019.

[4] C. Allen and T. Hospedales, "Analogies explained: Towards understanding word embeddings," in *International Conference on Machine Learning*, PMLR, 2019, pp. 223–231.

[5] R. M. Alvarez, *Computational social science*. Cambridge University Press, 2016.

[6] M. Antoniak and D. Mimno, "Evaluating the stability of embedding-based word similarities," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 107–119, 2018.

[7] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," in *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[8] B. Athiwaratkun, A. Wilson, and A. Anandkumar, "Probabilistic FastText for Multi-Sense Word Embeddings," in *ACL*, ACL, Jul. 2018, pp. 1–11.

[9] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[10] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *CoRR*, vol. abs/1803.01271, 2018. arXiv: 1803.01271.

[11] R. Bamler and S. Mandt, "Dynamic word embeddings," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 380–389.

[12] M. Baroni, G. Dinu, and G. Kruszewski, "Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, pp. 238–247.

[13] D. Beck, G. Haffari, and T. Cohn, "Graph-to-sequence learning using gated graph neural networks," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 273–283.

[14] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[15] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[16] W. Blacoe, E. Kashefi, and M. Lapata, "A quantum-theoretic approach to distributional semantics," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013, pp. 847–857.

[17] L. Bloomfield, "A set of postulates for the science of language," *Language*, vol. 2, no. 3, pp. 153–164, 1926.

[18] M. Born, "Zur Quantenmechanik der Sto\s svorgänge," *Zeitschrift für Physik*, vol. 37, no. 12, pp. 863–867, Dec. 1926, ISSN: 0044-3328. DOI: 10.1007/BF01397477.

[19] P. Bruza, K. Kitto, D. Nelson, and C. McEvoy, "Is there something quantum-like about the human mental lexicon?" *Journal of Mathematical Psychology*, vol. 53, no. 5, pp. 362–377, 2009.

[20] P. D. Bruza, K. Kitto, D. McEvoy, and C. McEvoy, "Entangling words and meaning," 2008.

[21] R. Cann, R. Kempson, and E. Gregoromichelaki, "Semantics: An introduction to meaning in language," 2009.

[22] Q. Chen, X. Zhu, Z.-H. Ling, S. Wei, H. Jiang, and D. Inkpen, "Enhanced lstm for natural language inference," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1657–1668.

[23] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[24] B. Coecke, G. de Felice, K. Meichanetzidis, and A. Toumi, "Foundations for near-term quantum natural language processing," *arXiv preprint arXiv:2012.03755*, 2020.

[25] T. Cohen and D. Widdows, "Bringing order to neural word embeddings with embeddings augmented by random permutations (earp)," in *Proceedings of the 22nd Conference on Computational Natural Language Learning*, 2018, pp. 465–475.

[26] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 670–680.

[27] A. Cucchiarelli, C. Morbidoni, G. Stilo, and P. Velardi, "A topic recommender for journalists," *Information Retrieval Journal*, vol. 22, no. 1, pp. 4–31, 2019.

[28] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[29] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 2978–2988.

[30] M. Davies, "Expanding horizons in historical linguistics with the 400-million word corpus of historical american english," *Corpora*, vol. 7, no. 2, pp. 121–157, 2012.

[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

[32] E. Di Buccio and M. Melucci, "Meeting and joining theme models in vector spaces for information retrieval," in *International Conference on Flexible Query Answering Systems*, Springer, 2017, pp. 59–70.

[33] ——, "Searching for information with meet and join operators," in *Quantum-Like Models for Information Retrieval and Decision-Making*, Springer, 2019, pp. 145–168.

[34] V. Di Carlo, F. Bianchi, and M. Palmonari, "Training temporal word embeddings with a compass," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 6326–6334, 2019. DOI: 10.1609/aaai.v33i01.33016326.

[35] H. Dubossarsky, S. Hengchen, N. Tahmasebi, and D. Schlechtweg, "Time-out: Temporal referencing for robust modeling of lexical semantic change," *ACL*, 2019.

[36] C. Fernando and S. Sojakka, "Pattern recognition in a bucket," in *Advances in Artificial Life*, W. Banzhaf, J. Ziegler, T. Christaller, P. Dittrich, and J. T. Kim, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 588–597, ISBN: 978-3-540-39432-7.

[37] J. R. Firth, "A synopsis of linguistic theory, 1930-1955," 1957.

[38] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[39] J. Gao, J.-Y. Nie, G. Wu, and G. Cao, "Dependence language model for information retrieval," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 2004, pp. 170–177.

[40] N. Garg, L. Schiebinger, D. Jurafsky, and J. Zou, "Word embeddings quantify 100 years of gender and ethnic stereotypes," *Proceedings of the National Academy of Sciences*, vol. 115, no. 16, E3635–E3644, 2018.

[41] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *International Conference on Machine Learning*, PMLR, 2017, pp. 1243–1252.

[42] A. M. Gleason, "Measures on the closed subspaces of a hilbert space," *Journal of mathematics and mechanics*, pp. 885–893, 1957.

[43] C. Goddard and A. Wierzbicka, *Semantic and lexical universals: Theory and empirical findings*. John Benjamins Publishing, 1994, vol. 25.

[44] A. Goyal and Y. Bengio, "Inductive biases for deep learning of higher-level cognition," *arXiv preprint arXiv:2011.15091*, 2020.

[45] P. R. Halmos, *Naive set theory*. Courier Dover Publications, 2017.

[46] P. R. Halmos and F.-D. V. Spaces, "Undergraduate texts in mathematics," *Finite-Dimensional Vector Spaces*, 1987.

[47] W. L. Hamilton, J. Leskovec, and D. Jurafsky, "Diachronic word embeddings reveal statistical laws of semantic change," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 1489–1501.

[48] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[49] H. He, K. Gimpel, and J. Lin, "Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks," in *EMNLP*, ACL, Sep. 2015, pp. 1576–1586.

[50] H. He and J. Lin, "Pairwise Word Interaction Modeling with Deep Neural Networks for Semantic Similarity Measurement," en, in *NAACL*, ACL, 2016, pp. 937–948. DOI: 10.18653/v1/N16-1108.

[51] F. Hill, K. Cho, and A. Korhonen, "Learning Distributed Representations of Sentences from Unlabelled Data," in *NAACL*, San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 1367–1377.

[52] F. Hill, K. Cho, A. Korhonen, and Y. Bengio, "Learning to Understand Phrases by Embedding the Dictionary," *TACL*, vol. 4, pp. 17–30, 2016, ISSN: 2307-387X.

[53] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[54] T. Hofmann, "Probabilistic latent semantic indexing," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 1999, pp. 50–57.

[55] Z. Hradil, J. Rehacek, J. Fiurasek, and M. Jezek, "3 Maximum-Likelihood Methodsin Quantum Mechanics," en, in *Quantum State Estimation*, ser. Lecture Notes in Physics, Springer, Berlin, Heidelberg, pp. 59–112.

[56] M. Hu and B. Liu, "Mining and Summarizing Customer Reviews," en, p. 10, 2014.

[57] S. Jain and B. C. Wallace, "Attention is not explanation," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 3543–3556.

[58] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, 2019.

[59] M. N. Jones, W. Kintsch, and D. J. Mewhort, "High-dimensional semantic space accounts of priming," *Journal of memory and language*, vol. 55, no. 4, pp. 534–552, 2006.

[60] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, Association for Computational Linguistics, Apr. 2017, pp. 427–431.

[61] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6769–6781.

[62] S. M. Kazemi, R. Goel, S. Eghbali, J. Ramanan, J. Sahota, S. Thakur, S. Wu, C. Smyth, P. Poupart, and M. Brubaker, "Time2vec: Learning a vector representation of time," *arXiv preprint arXiv:1907.05321*, 2019.

[63] Y. Kim, "Convolutional Neural Networks for Sentence Classification," en, *EMNLP*, pp. 1746–1751, 2014. DOI: 10.3115/v1/D14-1181.

[64] Y. Kim, Y.-I. Chiu, K. Hanaki, D. Hegde, and S. Petrov, "Temporal analysis of language through neural language models," *ACL*, p. 61, 2014.

[65] A. N. Kolmogorov, *Foundations of the theory of probability*, ser. Foundations of the theory of probability. Oxford, England: Chelsea Publishing Co., 1950, Pages: viii, 71.

[66] V. Kulkarni, R. Al-Rfou, B. Perozzi, and S. Skiena, "Statistically significant detection of linguistic change," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 625–635.

[67] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[68] T. K. Landauer and S. T. Dumais, "A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge.," *Psychological review*, vol. 104, no. 2, p. 211, 1997.

[69] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, pp. 255–258, 1995.

[70] Y. Lei, K. M. Hermann, P. Blunsom, and S. Pulman, "Deep learning for answer sentence selection," *Computer Science*, 2014.

[71] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14, Montreal, Canada: MIT Press, 2014, pp. 2177–2185.

[72] H. Li and J. Xu, "Semantic matching in search," *Found. Trends Inf. Retr.*, vol. 7, no. 5, pp. 343–469, Jun. 2014, ISSN: 1554-0669. DOI: 10.1561/1500000035.

[73] Q. Li, S. Uprety, B. Wang, and D. Song, "Quantum-Inspired Complex Word Embedding," in *Proceedings of The Third Workshop on Representation Learning for NLP*, Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 50–57.

[74] Q. Li, B. Wang, and M. Melucci, "Cnm: An interpretable complex-valued network for matching," *NAACL 2019. Best Explainable NLP Paper*, 2019.

[75] X. Li and D. Roth, "Learning Question Classifiers," in *COLING*, Association for Computational Linguistics, 2002.

[76] J. Lin, R. Nogueira, and A. Yates, "Pretrained transformers for text ranking: Bert and beyond," *arXiv preprint arXiv:2010.06467*, 2020.

[77] Z. C. Lipton, "The mythos of model interpretability," *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[78] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *arXiv preprint arXiv:2107.13586*, 2021.

[79] W. Liu, P. Zhou, Z. Zhao, Z. Wang, Q. Ju, H. Deng, and P. Wang, "K-bert: Enabling language representation with knowledge graph," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 2901–2908.

[80] L. Logeswaran and H. Lee, "An efficient framework for learning sentence representations," in *International Conference on Learning Representations*, 2018.

[81] R. Lorenz, A. Pearson, K. Meichanetzidis, D. Kartsaklis, and B. Coecke, "Qnlp in practice: Running compositional models of meaning on a quantum computer," *arXiv preprint arXiv:2102.12846*, 2021.

[82] K. Lund and C. Burgess, "Producing high-dimensional semantic spaces from lexical co-occurrence," *Behavior research methods, instruments, & computers*, vol. 28, no. 2, pp. 203–208, 1996.

[83] A. I. Lvovsky, "Iterative maximum-likelihood reconstruction in quantum homodyne tomography," *Journal of Optics B: Quantum and Semiclassical Optics*, vol. 6, no. 6, S556, 2004.

[84] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 1064–1074.

[85] M. Melucci, "A basis for information retrieval in context," *ACM Trans. Inf. Syst.*, vol. 26, no. 3, Jun. 2008, ISSN: 1046-8188. DOI: 10.1145/1361684.1361687.

[86] M. Melucci, *Introduction to Information Retrieval and Quantum Mechanics*. Springer, 2015, vol. 35.

[87] Y. Miao, L. Yu, and P. Blunsom, "Neural Variational Inference for Text Processing," *arXiv:1511.06038*, Nov. 2015.

[88] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations ofwords and phrases and their compositionality," *Advances in Neural Information Processing Systems*, vol. 26, pp. 3111–3119, 2013.

[89] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *ICLR*, 2013.

[90] D. Mimno and L. Thompson, "The strange geometry of skip-gram with negative sampling," in *Empirical Methods in Natural Language Processing*, 2017.

[91] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep learning–based text classification: A comprehensive review," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–40, 2021.

[92] B. B. Murdock, "A theory for the storage and retrieval of item and associative information.," *Psychological Review*, vol. 89, no. 6, p. 609, 1982.

[93] J. Neumann, *Mathematical foundations of quantum mechanics*. Princeton university press, 1955.

[94] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th. New York, NY, USA: Cambridge University Press, 2011.

[95] M. Pagliardini, P. Gupta, and M. Jaggi, "Unsupervised Learning of Sentence Embeddings Using Compositional n-Gram Features," en, *NAACL*, vol. 1, pp. 528–540, 2018. DOI: 10.18653/v1/N18-1049.

[96] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," en, Association for Computational Linguistics, 2005, pp. 115–124. DOI: 10.3115/1219840.1219855.

[97] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[98] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *NAACL*, 2018, pp. 2227–2237.

[99] A. Pinkus, "Weierstrass and approximation theory," *Journal of Approximation Theory*, vol. 107, no. 1, pp. 1–66, 2000.

[100] M. Pömsl and R. Lyapin, "Circe at semeval-2020 task 1: Ensembling context-free and context-dependent word representations," *ACL*, 2020.

[101] X. Qiu and X. Huang, "Convolutional neural tensor network architecture for community-based question answering.," in *IJCAI*, 2015, pp. 1305–1311.

[102] J. Řehá ček, Z. Hradil, and M. Je žek, "Iterative algorithm for reconstruction of entangled states," *Phys. Rev. A*, vol. 63, p. 040 303, 4 Mar. 2001. DOI: 10.1103/PhysRevA.63.040303.

[103] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, *Improving language understanding by generative pre-training*, 2018.

[104] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 2383–2392.

[105] S. Robertson and H. Zaragoza, *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.

[106] A. Rosenfeld and K. Erk, "Deep neural models of semantic shift," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 474–484.

[107] D. Rother, T. Haider, and S. Eger, "Cmce at semeval-2020 task 1: Clustering on manifolds of contextualized embeddings to detect historical meaning shifts," in *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, 2020, pp. 187–193.

[108] M. Rudolph and D. Blei, "Dynamic embeddings for language evolution," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 1003–1011.

[109] M. Sahlgren, A. Holst, and P. Kanerva, "Permutations as a means to encode order in word space," in *The 30th Annual Meeting of the Cognitive Science Society (CogSci'08), 23-26 July 2008, Washington DC, USA*, 2008.

[110] S. K. Sahu, F. Christopoulou, M. Miwa, and S. Ananiadou, "Inter-sentence relation extraction with document-level graph convolutional neural network," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 4309–4316.

[111] D. Schlechtweg, B. McGillivray, S. Hengchen, H. Dubossarsky, and N. Tahmasebi, "Semeval-2020 task 1: Unsupervised lexical semantic change detection," *SemEval*, 2020.

[112] S. Serrano and N. A. Smith, "Is attention interpretable?" In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 2931–2951.

[113] A. Severyn and A. Moschitti, "Learning to rank short text pairs with convolutional deep neural networks," in *International Acm Sigir Conference*, 2015.

[114] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2018, pp. 464–468.

[115] A. Shrivastava and P. Li, "Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips)," in *Advances in neural information processing systems*, 2014, pp. 2321–2329.

[116] R. Socher, C. C. Lin, C. Manning, and A. Y. Ng, "Parsing natural scenes and natural language with recursive neural networks," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 129–136.

[117] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[118] A. Sordoni, J. He, and J.-Y. Nie, "Modeling latent topic interactions using quantum interference for information retrieval," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2013, pp. 1197–1200.

[119] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[120] T. Szymanski, "Temporal word analogies: Identifying lexical replacement with diachronic word embeddings," in *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 2: short papers)*, 2017, pp. 448–453.

[121] N. Tahmasebi, L. Borin, and A. Jatowt, "Survey of computational approaches to lexical semantic change detection," *Computational approaches to semantic change*, pp. 1–91, 2021.

[122] D. Tang, B. Qin, and T. Liu, "Document Modeling with Gated Recurrent Neural Network for Sentiment Classification," in *EMNLP*, Lisbon, Portugal, Sep. 2015, pp. 1422–1432.

[123] Y. Tay, M. C. Phan, L. A. Tuan, and S. C. Hui, "Learning to rank question answer pairs with holographic dual lstm architecture," *arXiv preprint arXiv:1707.06372*, 2017.

[124] K. Tian, T. Zhang, and J. Zou, "Cover: Learning covariate-specific vector representations with tensor decompositions," in *International Conference on Machine Learning*, 2018, pp. 4926–4935.

[125] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal, "Deep complex networks," 2018.

[126] C. J. Van Rijsbergen, *The geometry of information retrieval*. Cambridge University Press, 2004.

[127] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[128] J. Von Neumann, *Mathematical foundations of quantum mechanics*, 2. Princeton university press, 1955.

[129] E. M. Voorhees and D. M. Tice, "Building a question answering test collection," *SIGIR*, pp. 200–207, Jul. 2000.

[130] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2018, pp. 353–355.

[131] B. Wang, E. Di Buccio, and M. Melucci, "Representing words in vector space and beyond," in *Quantum-Like Models for Information Retrieval and Decision-Making*, Springer, Cham, 2019, pp. 83–113.

[132] B. Wang, E. Di Buccio, and M. Melucci, "Sequential modeling in vector space," in *11th Italian Information Retrieval Workshop 2021*, 2021.

[133] B. Wang, E. Di Buccio, and M. Melucci, "Word2fun, modeling words as functions for dynamic word embeddings," in *NeurIPS 2021 (accepted)*, 2021.

[134] B. Wang, Q. Li, M. Melucci, and D. Song, "Semantic hilbert space for text representation learning," *WWW 2019*, 2019.

[135] B. Wang, L. Shang, C. Lioma, X. Jiang, H. Yang, Q. Liu, and J. G. Simonsen, "On position embeddings in bert," in *International Conference on Learning Representations*, vol. 2, 2021, pp. 12–13.

[136] B. Wang, P. Zhang, J. Li, D. Song, Y. Hou, and Z. Shang, "Exploration of quantum interference in document relevance judgement discrepancy," *Entropy*, vol. 18, no. 4, p. 144, 2016.

[137] B. Wang, D. Zhao, C. Lioma, Q. Li, P. Zhang, and J. G. Simonsen, "Encoding word order in complex embeddings," in *ICLR*, 2019.

[138] D. Wang and E. Nyberg, "A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering," in *ACL*, Jul. 2015, pp. 707–712.

[139] J. Wiebe, T. Wilson, and C. Cardie, "Annotating Expressions of Opinions and Emotions in Language," en, *Language Resources and Evaluation*, vol. 39, no. 2-3, pp. 165–210, May 2005, ISSN: 1574-020X, 1572-8412. DOI: 10.1007/s10579-005-7880-9.

[140] S. Wiegreffe and Y. Pinter, "Attention is not not explanation," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 11–20.

[141] M. Wolter and A. Yao, "Gated complex recurrent neural networks," *arXiv preprint arXiv:1806.08267*, 2018.

[142] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," en, *Nature*, vol. 299, no. 5886, pp. 802–803, Oct. 1982, ISSN: 1476-4687. DOI: 10.1038/299802a0.

[143] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[144] M. Xie, Y. Hou, P. Zhang, J. Li, W. Li, and D. Song, "Modeling quantum entanglements in quantum language models," 2015.

[145] C. Xing, D. Wang, C. Liu, and Y. Lin, "Normalized word embedding and orthogonal transform for bilingual word translation," in *NAACL*, 2015, pp. 1006–1011.

[146] L. Yang, Q. Ai, J. Guo, and W. B. Croft, "aNMM: Ranking Short Answer Texts with Attention-Based Neural Matching Model," in *CIKM*, ACM, 2016, pp. 287–296, ISBN: 978-1-4503-4073-1. DOI: 10.1145/2983323.2983818.

[147] Y. Yang, W.-t. Yih, and C. Meek, "WikiQA: A Challenge Dataset for Open-Domain Question Answering," in *EMNLP*, Association for Computational Linguistics, Sep. 2015, pp. 2013–2018.

[148] Z. Yao, Y. Sun, W. Ding, N. Rao, and H. Xiong, "Dynamic word embeddings for evolving semantic discovery," in *Proceedings of the eleventh acm international conference on web search and data mining*, 2018, pp. 673–681.

[149] W. Yin, H. Schütze, B. Xiang, and B. Zhou, "ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs," *Transactions of the ACL*, vol. 4, pp. 259–272, 2016, ISSN: 2307-387X.

[150] M. Young, "The stone-weierstrass theorem," in *MATH 328 Notes*, Queen's University at Kingston, 2006.

[151] A. W. Yu, D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le, "Qanet: Combining local convolution with global self-attention for reading comprehension," in *International Conference on Learning Representations*, 2018.

[152] L. Yu, K. M. Hermann, P. Blunsom, and S. Pulman, "Deep learning for answer sentence selection," in *NIPS Deep Learning and Representation Learning Workshop, Montreal*, 2014.

[153] C. Zhai and J. Lafferty, "A study of smoothing methods for language models applied to ad hoc information retrieval," in *ACM SIGIR Forum*, ACM New York, NY, USA, vol. 51, 2017, pp. 268–276.

[154] H. Zhang, M. Gu, X. Jiang, J. Thompson, H. Cai, S. Paesani, R. Santagati, A. Laing, Y. Zhang, M. Yung, *et al.*, "An optical neural chip for implementing complex-valued neural network," *Nature Communications*, vol. 12, no. 1, pp. 1–11, 2021.

[155] P. Zhang, J. Niu, Z. Su, B. Wang, L. Ma, and D. Song, "End-to-end quantum-like language models with application to question answering," 2018.

[156] J. Zhao, T. Wang, M. Yatskar, V. Ordonez, and K.-W. Chang, "Men also like shopping: Reducing gender bias amplification using corpus-level constraints," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.

[157] J. Zhao, Y. Zhou, Z. Li, W. Wang, and K.-W. Chang, "Learning gender-neutral word embeddings," in *EMNLP*, 2018.

[158] G. Zuccon, B. Piwowarski, and L. Azzopardi, "On the use of complex numbers in quantum models for information retrieval," in *Conference on the Theory of Information Retrieval*, Springer, 2011, pp. 346–350.