Complexifying BERT using LoRA Adapters

Fabio Tamburini[1]

[1]*FICLIT - University of Bologna, Via Zamboni, 32, Bologna, Italy*

Abstract
This paper presents the first results of a pilot study for transforming a real-valued pre-trained transformer encoder into a complex-valued one. Following recent findings about pre-training using LoRA, the main idea is to employ complex-valued LoRA adapters to make the trick and continue the pre-training of a given Italian model for setting up the adapters. After pre-training, the proposed complex-valued model has been evaluated on a standardised benchmark for Italian natural-language understanding obtaining very encouraging results.

Keywords
Complex-valued Transformers, Language-Model Pre-Training, LoRA Adapters, Evaluation, Italian

1. Introduction

The works from Arjovsky et al. [1], Trouillon et al. [2] and Trabelsi et al. [3] proposing complex-valued Deep Neural Networks (DNNs) rose an increasing interest on this type on Neural Networks for their intrinsic ability to manage problems defined on complex-valued features. For example, in the fields of signal and image processing, speech, signal and audio data are naturally complex-valued after Fourier, Laplace or Complex Wavelet transforms. Yang et al. [4] and Eilers and Jiang [5] presented state-of-the-art Automatic Music Transcription systems and Wang et al. [6] evaluated their complex-valued embeddings in text classification, machine translation and language modeling with promising results. Quantum-inspired Machine Learning, an emerging topic of research in NLP and AI, is completely based on complex-valued features and tensors. Liu et al. [7] presented a survey of novel quantum-cognitively inspired models that solved the task of sentiment analysis with good performances and Tamburini [8] proposed a Quantum WSD system based on static complex-valued embeddings obtained modifying the 'word2vec' [9] code.

The transformer encoder is a crucial component in transformer architectures [10]: primarily designed for processing input text and producing intermediate representations of input sequences, it consists of multiple layers of self-attention mechanisms and feed-forward neural networks, each contributing to the encoding process of both single words and entire sequences.

LoRA (Low-Rank Adaptation) [11] is a technique recently introduced to efficiently fine-tune transformer models. Instead of updating all the parameters of a large pre-trained model, LoRA introduces a small set of additional trainable parameters. These parameters are incorporated into the transformer layers through low-rank matrices, allowing the model to adapt to new tasks with significantly reduced computational and storage requirements. This method preserves the original model's performance while enabling quick and cost-effective customisation for specific applications.

A very recent work [12] suggested that, by applying LoRA adapters, it is possible to pre-train large transformer models from scratch obtaining comparable performance with respect to regular pre-training.

The main idea and contribution of this work consists in using LoRA adapters to convert a real-valued pre-trained transformer model into a complex-valued one being able to produce as output complex-valued word and sequence embeddings to be used in subsequent tasks. This process will require to continue the pre-training stage of a real-valued transformer model for setting up complex-valued LoRA adapters and train the global model to produce meaningful complex-valued embeddings.

Section 2 describes the state-of-the-art about complex-valued transformers; Section 3 presents the proposed model describing the internal details of our complex-valued LoRA-based transformer. Section 4 illustrates the obtained results when testing our complex-valued model on a benchmark for evaluating Natural Language Understanding (NLU) systems for the Italian Language [13] and Section 5 discusses the results and draws some conclusions.

2. Related Works

There are very few attempts in literature for creating a complex-valued transformer and all of them presuppose to pre-train the whole architecture from scratch, a very long and computationally demanding process, especially for large architectures.

CLiC-it 2024: Tenth Italian Conference on Computational Linguistics, Dec 04 — 06, 2024, Pisa, Italy
fabio.tamburini@unibo.it (F. Tamburini)
https://www.unibo.it/sitoweb/fabio.tamburini (F. Tamburini)
0000-0001-7950-0347 (F. Tamburini)

Yang et al. [4] concentrate on the development of a complex-valued transformer for speech, signal and audio data that are naturally complex-valued after Fourier Transform.

Wang et al. [6], working on positional embeddings and proposing a solution for modelling both the global absolute positions of words and their order relationships, introduced a small complex-valued transformer architecture to test their ideas.

The works from Eilers and Jiang [5] and Li et al. [14] have the goal of providing a complete model for building complex-valued transformer encoders, describing possible building blocks for doing it, testing different configurations and parameters.

As we said before, all these works pre-train their proposal from scratch and none of them proposed to use adapters as we will describe in the next section.

3. The Proposed Model

The starting point for our work is the BERT model. BERT (Bidirectional Encoder Representations from Transformers) is a language representation model introduced by Google in 2018. It is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers, making it deeply bidirectional.

Even if the present work is devoted to "complexify" the BERT architecture for Italian, all the steps presented in the following sections can be used for any pre-trained version of BERT in different languages. Moreover, these steps forms, in principle, building blocks to complexify any transformer architecture.

3.1. Complex Numbers

Complex numbers are an extension of the real number system. They consist of two parts: a real part and an imaginary part. The imaginary part is defined using the imaginary unit i, where $i^2 = -1$. A complex number is typically written in the form $c = a + bi$, where a and b are real numbers. Given c, $\mathcal{R}(c)$ and $\mathcal{I}(c)$ return, respectively, the real and imaginary part of c.

The development of complex numbers allows for a more complete understanding of algebraic equations, especially those that have no real solutions and are crucial in various fields such as engineering, physics, and applied mathematics, providing tools for analysing waveforms, electrical circuits, and quantum mechanics.

All the standard algebraic operations on real numbers can be extended or defined also on the complex field $\mathbb{C}$. Moreover, the complex conjugate of a complex number is obtained by changing the sign of its imaginary part. For a complex number $c = a + bi$ its complex conjugate is $\bar{c} = a - bi$. In the context of matrices, the conjugate transpose (also known as the Hermitian transpose) involves taking the transpose of a matrix and then taking the complex conjugate of each element; given a complex-valued matrix A, it is usually denoted as $A^\dagger$.

3.2. LoRA Adapters

When fine-tuning a pre-trained language model, the goal is to adjust the model parameters to better fit a specific task. However, large language models have millions or billions of parameters, making this process resource-intensive. LoRA [11] addresses this by introducing a low-rank decomposition approach to fine-tuning.

Suppose we have a pre-trained model with weight matrices W in various layers. For simplicity, consider a single weight matrix $W \in \mathbb{R}^{n \times m}$. LoRA approximates the update to the weight matrix ΔW using a low-rank factorization. Instead of directly updating W, as $W' = W + \Delta W$, we decompose the update as $\Delta W = A \cdot B^T$, where $A \in \mathbb{R}^{m \times r}$ and $B \in \mathbb{R}^{n \times r}$, with $r \ll min(m, n)$. A and B are the learnable parameters, while W usually remains fixed.

LoRA adapters provide an efficient method for fine-tuning large models by leveraging low-rank approximations. This approach reduces the number of trainable parameters and computational cost while maintaining the model's performance, making it a practical solution for adapting large-scale pre-trained models to specific tasks.

Moreover, Lialin et al. [12] showed that we can safely apply LoRA also for pre-training transformer encoders from scratch obtaining performances comparable to the original models.

Given these premises, the main idea introduced by this work is to define A and B as complex-valued matrices used to adapt a generic weight matrix W of the pre-trained real-valued model to produce complex-valued outputs. All the W matrices will be kept frozen and the standard LoRA forward update with input vector x will become $y = (W + A \cdot B^\dagger)\, x$.

3.3. Embeddings

The BERT embedding layer is responsible for converting input tokens into dense vectors that can be processed by subsequent layers. It consists of three main components, the Token Embeddings, that map each token to a fixed-size vector representation, the Segment Embeddings, that add a segment identifier to each token to distinguish between different segments (e.g., sentences) and the Positional Embeddings that mark positional information to capture the order of tokens. These three embeddings are learned during the pre-training phase and summed to form the final input embedding, which is then passed to the transformer encoder layers for further processing.

Each component represents the corresponding embeddings as a real-valued matrix that can be made complex-valued by summing a complex-valued LoRA adapter as described in Section 3.2.

3.4. Multi-head Self-Attention

Self-attention is a mechanism in neural networks that allows each element of an input sequence to focus on, or "attend to", other elements in the same sequence. In the context of BERT and other transformer models, self-attention helps capture the relationships and dependencies between words, regardless of their distance from each other in the text.

The self-attention mechanism can be succinctly expressed in matrix form as:

$$Q = X \cdot W^Q, \;\; K = X \cdot W^K, \;\; V = X \cdot W^V$$
$$Attention(Q, K, V) = softmax\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V$$

where $X \in \mathbb{R}^{d \times n}$ is the input embedding matrix, $W^Q, W^K, W^V \in \mathbb{R}^{d \times d_k}$ are projection matrices, d is the input embedding size and $d_k = d/\#heads$. The output matrix, once concatenated the contributions of the different heads and further projected into the initial dimension d, contains the context-aware representations for each word in the input sequence, incorporating information from all other words as determined by their relevance.

In order to convert the real-valued self-attention mechanism to manage complex-valued inputs, it is sufficient to modify the three projections matrices W^Q, W^K, W^V using a complex-valued LoRA adapter as shown before and modify the attention computation as

$$Attention(Q, K, V) = softmax\left(\frac{|Q \cdot K^\dagger|}{\sqrt{d_k}}\right) \cdot V$$

The complex-valued Query and Key vectors are then multiplied and the modulus of each complex-valued component for the resulting vector is computed (as suggested in Eilers and Jiang [5], Li et al. [14]), normalised by $\sqrt{d_k}$ and transformed into a probability distribution by the softmax function to be used as attention vector for the complex-valued vector V.

3.5. Linear Layers

A linear layer, also known as a fully connected layer or dense layer, is a fundamental building block in transformer networks. It performs a linear transformation on the input data by applying a weight matrix and adding a bias vector. Mathematically, it can be described as $Output = x \cdot W + b$, where x is the input vector, W the weight matrix and b the bias vector.

As before, to tranform a real-valued linear layer into a complex-valued one, it is sufficient to apply a LoRA adapter to the weight matrix and add a further complex-valued bias vector z to the result, mathematically:

$$Output = x \cdot (W + A \cdot B^\dagger) + (b + z).$$

3.6. Complex Layer Normalisation

As suggested in Eilers and Jiang [5], Li et al. [14], normalising real and imaginary parts separately could lead to poor normalisations and very elliptical distributions. Inspired by the work of Eilers and Jiang [5], we normalised a generic complex vector $z \in \mathbb{C}$ by first computing

$$E(z) = \frac{1}{n}\sum_{j=1}^{n} z_j$$
$$Cov_{\mathbb{C}}(z) = \begin{pmatrix} Var(\mathcal{R}(z)) & Cov(\mathcal{R}(z), \mathcal{I}(z)) \\ Cov(\mathcal{R}(z), \mathcal{I}(z)) & Var(\mathcal{I}(z)) \end{pmatrix}$$

where Var and Cov indicate the real-valued Variance and Covariance functions, and then produce a normalised output vector

$$z' = u \cdot \sqrt{Cov_{\mathbb{C}}^{-1}(z)} \cdot \begin{pmatrix} \mathcal{R}(z - E(z)) \\ \mathcal{I}(z - E(z)) \end{pmatrix} + v$$

where u and v are two vectors of the same dimension of z for applying an affine transformation to the normalised vector.

3.7. Activation Function

In BERT, the primary activation function used is the Gaussian Error Linear Unit (GELU). We extended this function to complex-valued inputs in a simple way following Li et al. [14] as:

$$SplitGELU(z) = GELU(\mathcal{R}(z)) + i\, GELU(\mathcal{I}(z))$$

where $z \in \mathbb{C}^n$ is a generic complex-valued vector. With regard to the pooling layer, we applied the same principle to the $\tanh$ activation.

3.8. Training Heads and Loss Functions

In BERT, the term "training heads" refers to the additional layers added on top of the base BERT model for solving specific tasks. These heads are tailored to the type of problem BERT is being fine-tuned to solve. The most common training heads include the Masked Language Model (MLM) and Next Sentence Prediction (NSP) heads used for BERT pre-training and Sequence/Token Classification heads trained alongside the base BERT model during fine-tuning, enabling the model to be adapted to various NLU tasks by leveraging its robust contextual embeddings.

In the proposed model, all these training heads are configured in the same way as a single LoRA-adapted linear layer, as described in Section 3.5, applying the modulus function for transforming the complex-valued output into a real-valued one and inject it into a standard real-valued Cross Entropy loss function.

4. Experiments

All the experiments presented in this work rely on the same base Italian BERT model used as baseline in Basile et al. [13], namely "*dbmdz/bert-base-italian-xxl-uncased*" (abbreviated as 'ItalianBERT_XXL' as in the cited paper), available in the Huggingface model repository[1].

4.1. Datasets for Pre-training and Evaluation

Pre-Training. The dataset we used for continuing the pre-training of the proposed model in order to set up the complex-valued LoRA parameters is similar to that used for pre-training the basic model from DBMDZ. It is formed by the 1/3/2022 dump of the Italian Wikipedia available on the Huggingface datasets repository and an equivalent "BookCorpus" we built using Italian ebooks.

During the pre-training phase we adopted the same hyperparameters used for training BERT, namely a learning rate of 1e-4, with a linear schedule with warmup, and a batch size of 512.

Evaluation. The performance evaluation for the proposed complex-valued model has been performed by relying on the Unified Interactive Natural Understanding of the Italian Language (UINAUIL) dataset collection, a benchmark of six tasks for Italian Natural Language Understanding [13]. Table 1 lists the datasets contained in UINAUIL with a short task description and datasets dimensions.

It is important to clarify that the goal of this work is not to produce a powerful model for achieving the best scores in the leaderboards, but instead we relied on a standardised dataset to verify if our complex-valued model is able to produce reliable embeddings that can be used for solving downstream tasks through fine-tuning exhibiting similar performances with standard real-valued models (in this case, the cited 'ItalianBERT_XXL').

All models has been fine-tuned for exactly 2 epochs, with a learning rate of 1e-4, as in the cited experiments from Basile et al. [13] and with a batch size of 32 (unique exception the task TE that did not converge with a batch size bigger than 4).

[1]https://huggingface.co/dbmdz/bert-base-italian-xxl-uncased

4.2. Results

The influential paper from Reimers and Gurevych [15] makes clear to the community that reporting a single score for each DNN training/evaluation session could be heavily affected by the system random initialisation and we should instead report the mean and standard deviation of various runs, with the same setting, in order to get a more accurate picture of the real systems performance and make more reliable comparisons between them. For these reasons, any result proposed in this paper is presented as the mean and standard deviation of the relevant metric over 5 runs with different random initialisations. We have also recomputed, using the same protocol, the baseline results from Basile et al. [13] and introduced a further baseline that always assigns the highest frequency class.

Table 2 shows the number of parameters for all the models tested in our experiments, split between trainable and non-trainable.

Table 3 shows the performance results of the various models in solving the UINAUIL tasks: our proposed models exhibit performances in line with the original model and sometimes better, especially for small-to-mid LoRA ranks, with r equal to 16, 32 and 64.

5. Discussion and Conclusions

In our evaluation experiments we adopted the hyperparameters proposed in Basile et al. [13] for maintaining comparability, but our models are bigger and more complex and, maybe, need more training epochs and/or different learning rates to achieve a full convergence during the fine-tuning phase for evaluation. For example, we were forced to reduce the learning rate to 1e-5 for each model evaluated on TE benchmark to favour convergence. Again, we clarify that the goal of this work is not to beat other systems in the leaderboards, but to show the effectiveness of this approach for complexifying transformer architectures and we think that the results confirm our initial research question.

Having complexified BERT matrices by adding LoRA adapters, we have no guarantee, in principle, that the system will not converge to the original BERT-based model setting all adapters to zero and nullify all imaginary part in the complex-valued model. We checked this in various ways and, as shown in Figure 1, some randomly chosen complex-valued components of token embeddings for the CmplxBERTLoRA_16 model show to cover the entire complex space in a uniform way, supporting the idea that

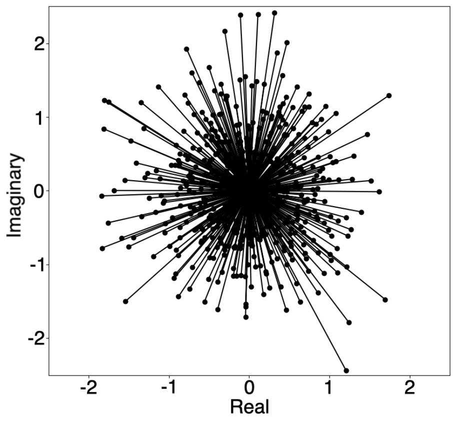
Table 1
Summary of the tasks included in UINAUIL from [13].

Acronym	Full name	Task type	Size (training/test)
TE	Textual Entailment	Sentence pair classification	400/400
EVENTI	Event detection & classification	Sequence labeling	5,889/917
FactA	Factuality classification	Sequence labeling	2,723/1,816
SENTIPOLC	Sentiment Polarity Classification	Sentence classification	7,410/2,000
IronITA	Irony Detection	Sentence classification	3,777/872
HaSpeeDe	Hate Speech Detection	Sentence classification	6,839/1,263

Table 2
Number of parameters for the different models tested in this work. With regard to the complex-valued BERT - 'CmplxBERT-LoRA' - the number at the end of the name indicates the LoRA rank r and the first column the complex-valued LoRA parameters trained during the continuation of the pre-training phase.

Model	Trainable	Non-Trainable	Total
ItalianBERT_XXL	111.3M	-	111.3M
CmplxBERTLoRA_8	3.5M	111.3M	114.8M
CmplxBERTLoRA_16	6.8M	111.3M	118.1M
CmplxBERTLoRA_32	13.3M	111.3M	124.6M
CmplxBERTLoRA_64	26.4M	111.3M	137.7M
CmplxBERTLoRA_128	52.6M	111.3M	163.9M

Figure 1: Argand diagram of some randomly chosen components for the complex-valued token embeddings computed for a sample sentence by the CmplxBERTLoRA_16 model.



the pre-training phase consistently adapted the starting real-valued model to produce reliable complex-valued embeddings.

We did also some experiments with a real-valued LoRA model containing about the same number of parameters of CmplxBERTLoRA_8, adding real-valued adapters of rank 16, to investigate if a complex-valued transformer is able to produce better results that an equivalent real-valued one, but such experiments did not show any relevant performance differences between the two models.

This work presented a relevant set of experiments for testing the idea of being able to complexify a Transformer encoder architecture like BERT by using complex-valued LoRA adapters. The obtained results on Italian models are very encouraging showing in a clear way that this technique is effective in transforming a real-valued pre-trained model into a complex-valued one maintaining the same level of performance.

We have to say that the UINAUIL benchmark is not without problems: TE dataset is very small and such large models struggle to reliably converge to a reasonable minimum during training leading to very unstable results. FactA is very problematic as well: classes are strongly skewed and the Max_Freq_Baseline, always choosing the highest-frequency class, is able to achieve an accuracy of 0.967! For all these reasons, we think that these two benchmarks should be excluded from any real evaluation.

This pilot study presents only the first step for proposing building blocks based on LoRA adapters for complexifying any kind of transformer, either for representation learning or for text generation or for both processes together. All the complex-valued models were pre-trained on various GPUs for speeding up the experiments, but a general CmplxBERTLoRA model can be trained on a single 12/16GB GPU without problems, while the pre-training of a complex-valued BERT model from scratch would have required at least 4 NVIDIA A100 64GB GPUs for obtaining results in reasonable time. Using LoRA for 'complexifying' a model mitigates the need of complex and expensive computational infrastructures not easily available to any scholar.

Code and models are available on github[2].

[2]https://github.com/ftamburin/CmplxBERTLoRA

Table 3
Experiments results when testing the considered models on the UIANUIL tasks, presented as mean and standard deviation of 5 runs. The official metric is marked with an arrow pointing in the direction of the best values. The best result for each task is marked in boldface while the underlined value is the best result obtained by our complex-valued model.

	TE				SENTIPOLC				EVENTI
Model	**P**	**R**	**F1↑**	**Acc.**	**P**	**R**	**F1↑**	**Acc.**	**Acc.↑**
Max_Freq_Baseline	.275	.500	.355	.550	.360	.500	.416	.457	.839
ItalianBERT_XXL [13]	.391	.495	.379	.541	.764	.741	.740	.675	.936
ItalianBERT_XXL	.524	.502	.383	.548	.758	.732	.733	.663	**.958**
(recomputed by us)	±.0608	±.0039	±.0267	±.0045	±.0051	±.0066	±.0081	±.0123	±.0002
CmplxBERTLoRA_8	.680	.540	.453	.583	.764	.748	.747	.680	.957
	±.0548	±.0222	±.0540	±.0176	±.0107	±.0069	±.0072	±.0068	±.0006
CmplxBERTLoRA_16	.627	.538	.459	.580	.766	.747	**.750**	.685	.957
	±.0260	±.0166	±.0369	±.0135	±.0125	±.0059	±.0079	±.0093	±.0003
CmplxBERTLoRA_32	.667	.597	**.551**	.627	.762	.741	.742	.675	.957
	±.0225	±.0698	±.1225	±.0550	±.0065	±.0068	±.0071	±.0061	±.0012
CmplxBERTLoRA_64	.652	.569	.509	.606	.761	.745	.743	.674	**.958**
	±.0360	±.0528	±.0894	±.0441	±.0090	±.0102	±.0106	±.0120	±.0007
CmplxBERTLoRA_128	.613	.561	.514	.592	.750	.733	.729	.657	.957
	±.0641	±.0555	±.0912	±.0511	±.0121	±.0107	±.0152	±.0199	±.0013

	IronITA				HaSpeeDe				FactA
Model	**P**	**R**	**F1↑**	**Acc.**	**P**	**R**	**F1↑**	**Acc.**	**Acc.↑**
Max_Freq_Baseline	.249	.500	.333	.499	.254	0.500	.337	.508	**.967**
ItalianBERT_XXL [13]	.769	.765	.764	.765	.792	.791	.791	.791	.908
ItalianBERT_XXL	.772	.769	**.769**	.769	.790	.789	.788	.788	.911
(recomputed by us)	±.0098	±.0101	±.0102	±.0101	±.0122	±.0154	±.0165	±.0159	±.0022
CmplxBERTLoRA_8	.750	.746	.745	.746	.787	.784	.783	.783	.909
	±.0101	±.0089	±.0090	±.0089	±.0040	±.0064	±.0071	±.0066	±.0028
CmplxBERTLoRA_16	.754	.751	.751	.751	.780	.778	.777	.777	.907
	±.0075	±.0061	±.0060	±.0061	±.0076	±.0073	±.0072	±.0073	±.0028
CmplxBERTLoRA_32	.750	.747	.746	.747	.794	.790	**.789**	.789	.907
	±.0119	±.0095	±.0090	±.0095	±.0117	±.0132	±.0139	±.0135	±.0022
CmplxBERTLoRA_64	.755	.753	.752	.753	.789	.785	.784	.784	.910
	±.0048	±.0040	±.0038	±.0039	±.0081	±.0106	±.0115	±.0111	±.0012
CmplxBERTLoRA_128	.744	.741	.741	.742	.785	.779	.777	.778	.909
	±.0176	±.0178	±.0180	±.0176	±.0116	±.0134	±.0142	±.0137	±.0031

References

[1] M. Arjovsky, A. Shah, Y. Bengio, Unitary evolution recurrent neural networks, in: Proceedings of the 33rd International Conference on International Conference on Machine Learning - ICML'16, JMLR.org, 2016, p. 1120–1128.

[2] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: Proceedings of the 33rd International Conference on International Conference on Machine Learning - ICML'16, JMLR.org, 2016, p. 2071–2080.

[3] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, C. J. Pal, Deep Complex Networks, in: Proc. of the International Conference on Learning Representations, ICLR 2018, 2018.

[4] M. Yang, M. Q. Ma, D. Li, Y.-H. H. Tsai, R. Salakhutdinov, Complex Transformer: A Framework for Modeling Complex-Valued Sequence, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020), 2020, pp. 4232–4236.

[5] F. Eilers, X. Jiang, Building Blocks for a Complex-Valued Transformer Architecture, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE Signal Processing Society, 2023.

[6] B. Wang, D. Zhao, C. Lioma, Q. Li, P. Zhang, J. G. Simonsen, Encoding word order in complex embeddings, in: Proceedings of the International Conference on Learning Representations, 2020.

[7] Y. Liu, Q. Li, B. Wang, Y. Zhang, D. Song, A survey of quantum-cognitively inspired sentiment analysis models, ACM Comput. Surv. 56 (2023).

[8] F. Tamburini, A quantum-like approach to word sense disambiguation, in: R. Mitkov, G. Angelova (Eds.), Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019), INCOMA Ltd., Varna, Bulgaria, 2019, pp. 1176–1185.

[9] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: C. Burges, et al. (Eds.), Advances in Neural Information Processing Systems 26, Curran Associates, Inc., 2013, pp. 3111–3119.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 30, Curran Associates, Inc., 2017.

[11] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, LoRA: Low-rank adaptation of large language models, in: Proceedings of the International Conference on Learning Representations, 2022.

[12] V. Lialin, N. Shivagunde, S. Muckatira, A. Rumshisky, ReLoRA: High-Rank Training Through Low-Rank Updates, in: Proceedings of the International Conference on Learning Representations, Vienna, Austria, 2024.

[13] V. Basile, L. Bioglio, A. Bosca, C. Bosco, V. Patti, UINAUIL: A unified benchmark for Italian natural language understanding, in: D. Bollegala, R. Huang, A. Ritter (Eds.), Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), Association for Computational Linguistics, Toronto, Canada, 2023, pp. 348–356.

[14] Q. Li, B. Wang, Y. Zhu, C. Lioma, Q. Liu, Adapting Pre-trained Language Models for Quantum Natural Language Processing, 2023. arXiv:2302.13812.

[15] N. Reimers, I. Gurevych, Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, ACL, Copenhagen, Denmark, 2017, pp. 338–348.