11th International Conference on Electrical, Electronics and Computer Engineering (IcETRAN), Nis, 3-6 June 2024

Traditional and parameter-efficient fine-tuning of LLMs for sentiment analysis in the English and Serbian language

Nikola Đorđević
Department of computer science
Faculty of electronic engineering, University of Niš
Niš, Serbia
nikola.djordjevic@elfak.ni.ac.rs, ORCID: 0009-0009-8662-4079

Suzana Stojković
Department of computer science
Faculty of electronic engineering, University of Niš
Niš, Serbia
suzana.stojkovic@elfak.ni.ac.rs, ORCID: 0000-0003-1111-417X

***Abstract*—Fine-tuning of large language models is often demanding in terms of computational resources and memory. Consequently, there is a need to explore new methods that can effectively fine-tune these models without compromising performance. In this paper, we compared traditional and parameter-efficient fine-tuning of large language model XLM-RoBERTa, across both high-resourced languages like English and low-resourced languages such as Serbian. The model is fine-tuned for sentiment analysis when it is conducted using text classification. The experiments have shown that parameter-efficient fine-tuning can produce results that are on par with traditional fine-tuning, both for the English and the Serbian language. Furthermore, parameter efficient fine-tuning approach significantly reduces the computational time and memory requirements associated with model training, thereby offering a promising avenue for resource-efficient large language models fine-tuning. In experiments with the English language, we achieved accuracy using the PEFT approach that was only 3% lower than the accuracy obtained through the traditional approach, while reducing training time by 35% and memory requirements by 42%. In experiments involving the Serbian language, we achieved accuracy using the PEFT approach that was only 1% lower than the accuracy obtained through the traditional approach, while reducing training time by 42% and memory requirements by 69%.**

Keywords—parameter-efficient fine-tuning, sentiment analysis, large language models

I. Introduction

Large language models are starting to represent the standard when it comes to Natural Language Processing (NLP). However, training and using these models can be very memory and time-intensive, especially considering that the number of parameters of these models is constantly increasing. These models are frequently inaccessible on standard personal computers. Therefore, there is a need to find optimal ways to train and use large language models. The goal of this work is to examine a relatively new approach in the training of LLMs – parameter-efficient fine-tuning (PEFT) [1] and compare it with traditional fine-tuning. The second objective is to compare the performance of the fine-tuned models for a globally used language, such as English, and for a rarely used language, such as Serbian. The assumption is that the effectiveness of PEFT may differ for the two languages.

The difference should be made when dealing with NLP tasks in a language that is rarely used, such as Serbian, and a language that is globally used, like English. When dealing with NLP tasks in low-resourced languages, we face various challenges, unlike languages that are used globally. These challenges arise because there are not enough corpora, data preprocessing tools, and pre-trained models for the low-resourced languages. An additional problem with the Serbian language is its complexity.

The NLP task we chose to compare these two approaches in model fine-tuning is sentiment analysis. Sentiment analysis automates the detection, extraction, and classification of emotions, attitudes, and opinions in text. There are different approaches to implementing sentiment analysis – by using text classification methods or by using sentiment lexicons. In this paper, a text classification method is used.

In this paper, we fine-tuned the multilingual XLM-RoBERTa [2] model for sentiment analysis in English and in the Serbian language. We used SentiComments.SR corpus [3] for the Serbian language and IMBD dataset for the English language [4]. We conducted traditional fine-tuning and fine-tuning with PEFT. Parameter-efficient fine-tuning was performed using the Low-Rank Adaptation technique (LORA) [5].

In continuation, the paper is organized as follows: the second section introduces used models and fine-tuning approaches. The next chapter describes the various ways in which the system is implemented and the results gathered in experiments. Finally, in the concluding section, all experiments are summarized and directions for further research on this topic are proposed.

II. Large Language Models and Fine-Tuning Approaches

Large language models are based on transformer architecture. Transformer is a deep neural network consisting of encoder and decoder parts [6]. The encoder maps the input sequence of symbols to a sequence of continuous representations. The decoder takes that sequence of continuous representations as input and generates a sequence of symbols, one element at a time. The model takes the previously generated symbols as additional input when generating the next. The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and the decoder. The key component of transformer-based models is a self-attention mechanism. The self-attention enables the model to learn long-range dependencies and take better context when generating continuous representations of input sequences. Transformers are more parallelizable and require significantly less time to train [6]. The use of transformer models is based on transfer learning, which means that the model, pre-trained on general-purpose tasks, is fine-tuned on specific tasks for our needs.

BERT [7] and GPT [8] were the first LLMs that captured significant attention. BERT focuses on bidirectional contextual understanding and is very good for NLP tasks such as text classification, named entity recognition, semantic similarity, and so on. On the other hand, GPT models focus on generative capabilities and are excellent at text generation. Several other notable LLMs have garnered attention, including T5, BART, RoBERTa, and XLNet, among others. LLMs with an enormous number of parameters are becoming more and more popular, such as Llama2, Falcon, Gemini, GPT-4, and so on.

The LLM we used is multilingual XLM-RoBERTa in base size. We chose the multilingual model because we performed sentiment analysis both in the English and the Serbian language. Standard RoBERTa is presented in [9]. It has been observed that BERT is significantly undertrained and a new approach for training BERT models, called RoBERTa involved prolonging the training duration, utilizing larger batches, incorporating more extensive datasets, eliminating the next sentence prediction objective, extending the sequence length during training, and dynamically altering the masking pattern applied to the training data. XLM-RoBERTa is a multilingual extension of RoBERTa [2]. Trained on text from 100 languages, XLM-RoBERTa exhibits state-of-the-art performance across various tasks, including cross-lingual classification, sequence labeling, and question answering.

Large language models are pre-trained on a large corpus of data and have general knowledge. To maximize the capabilities of LLMs, we have to fine-tune them on task-specific data to get better performance on downstream tasks. Traditional fine-tuning of models involves training the entire model, including all layers and parameters. While some layers may be frozen, this often yields bad results. Frequently, training all layers of the model is necessary to achieve the best performance on downstream tasks. Training the entire model can be highly demanding in terms of computational resources and time, especially for models with an enormous number of parameters. It can be assumed that the future of training and using large language models will rely on finding optimal methods for training and utilization, making these models accessible even on average computers. In this paper, we focus on parameter-efficient fine-tuning.

PEFT employs a range of deep learning techniques to diminish the number of trainable parameters while preserving performance comparable to full fine-tuning [10]. Moreover, PEFT selectively updates a small fraction of additional parameters or modifies a subset of the pre-trained parameters, thus saving the general knowledge of LLM. Additionally, since the size of the fine-tuned dataset is typically much smaller than the pre-trained dataset, conducting full fine-tuning to update all pre-trained parameters may result in overfitting. PEFT addresses this potential problem by selectively updating pre-trained parameters.

Parameter-efficient fine-tuning can be executed through various techniques. Some of these techniques include prompt tuning, prefix tuning, utilizing adapters, Low-Rank Adaptation techniques, and so on. In hard prompt tuning, efforts are made to provide the model the best prompt to optimize its performance on the desired task. In soft prompt tuning, the entire pre-trained model is frozen, and only a limited number of additional tunable tokens per downstream task are allowed to be prepended to the input text. Tuning with adapter modules entails incorporating a small set of new parameters into a model, which are then trained specifically for the downstream task [1]. In conventional fine-tuning of deep networks, adjustments are typically made solely to the top layer of the network. However, this is necessary because the label spaces and losses between the upstream and downstream tasks often vary. Adapter modules, on the other hand, facilitate more general architectural adjustments to repurpose a pre-trained network for a downstream task. The adapter fine-tuning approach involves integrating new layers into the original network.

In this paper, we used Low-Rank Adaptation (LoRA) [5] as one of the PEFT techniques. The core principle of LoRA involves representing weight updates with two smaller matrices obtained through low-rank decomposition. These matrices are trainable to adapt to new data while minimizing overall modifications. Notably, the original weight matrix remains intact and undergoes no further adjustments. The final outcomes are achieved by integrating both the original and adapted weights. Using LoRA offers several advantages. Firstly, it significantly improves fine-tuning efficiency by reducing the number of trainable parameters. Moreover, LoRA is compatible with various other parameter-efficient techniques and can be integrated with them seamlessly. Models fine-tuned with LoRA exhibit performance on par with fully fine-tuned models. Also, LoRA doesn't introduce any additional inference latency, as adapter weights can be smoothly merged with the base model.

III. Implementation and Experimental Results

The dataset used for sentiment analysis in the English language is the IMBD movie reviews dataset [4]. This dataset is intended for binary sentiment classification and contains 25,000 highly polar movie reviews for training and 25,000 for testing. We concatenated these two datasets in one and then selected 3490 instances with the aim that the number of instances is equal to the number of instances we use for the Serbian language. In this reduced dataset, half of the instances belong to the positive and half belong to the negative class.

The dataset used for sentiment analysis in the Serbian language is SentiComments.SR corpus [3]. This corpus contains reviews from the site "Kakavfilm" written in Serbian. There are 6 labels in the dataset:

- +1 – for texts that are entirely or predominantly positive
- -1 – for texts that are entirely or predominantly negative
- +M – for texts that convey an ambiguous sentiment or a mixture of sentiments, but lean more towards the positive sentiment in a strict binary classification
- -M – for texts that convey an ambiguous sentiment or a mixture of sentiments, but lean more towards the negative sentiment in a strict binary classification
- +NS – for texts that only contain non-sentiment-related statements, but still lean more towards the positive sentiment in a strict binary classification
- -NS – for texts that only contain non-sentiment-related statements, but still lean more towards the negative sentiment in a strict binary classification.

TABLE I. RESULSTS OF EXPERIMENTS IN THE ENGLISH LANGUAGE

Trainable %	Time	Memory	Epochs	Accuracy
100	11m 5s	10139MiB	5	0.8885
100	22m 15s	10139MiB	10	0.8850
100	44m 9s	10139MiB	20	0.8746
0.42329481296123295	7m 11s	5845MiB	5	0.8606
0.42329481296123295	14m 21s	5845MiB	10	0.8780
0.42329481296123295	28m 43s	5846MiB	20	0.8920
1.050342201278925	7m 14s	5885MiB	5	0.8571
1.050342201278925	14m 29s	5885MiB	10	0.8780
1.050342201278925	28m 58s	5885MiB	20	0.8990
1.8742231413082096	7m 19s	5951MiB	5	0.8606
1.8742231413082096	14 40s	5951MiB	10	0.8780
1.8742231413082096	29m 17s	5951MiB	20	0.8990

Initially, we grouped all instances in three classes, so that +1 and +M belong to positive, -1 and -M to negative, and +NS and -NS belong to neutral sentiment. Then, we removed neutral instances and obtained a dataset with two classes – positive and negative. In this dataset, about 60% of instances belong to positive and about 40% belong to negative class. The total number of instances is 3490.

All models were instantiated using the AutoModelForSequenceClassification class and trained employing the Trainer class. Both classes are available in the transformers library [11]. Parameter-efficient fine-tuning with Low-rank adaptation was implemented using the LoraConfig class from the peft library [12]. The batch size in all experiments was set to 8. The learning rate was set to 2e-5, with warmup steps of 500 and weight decay of 0.01. For each specific case involving a different percentage of trained parameters, we experimented with varying numbers of epochs – 5, 10, and 20. All experiments were done on Nvidia GeForce RTX 4060 GPU with memory of 16GB.

Table I shows the results of experiments with text in the English language. In addition to assessing the model's performance in terms of accuracy, we also monitored resource utilization, including memory usage and training time. We can see that our initial experiment, which included fine-tuning all parameters of the XLM-RoBERTa model in 5 epochs, achieved a good accuracy of 88 percent. The process of training the model itself did not take so much time and took up more than 50% percent of graphic card memory. With increasing the number of epochs to 10, the accuracy did not change but the time required for training doubled. With increasing the number of epochs to 20 the accuracy decreased slightly but the training time became even four times higher compared to the first experiment with 5 epochs. We can assume that the model started to overfit with an increased number of epochs. Taking everything into account, when training all model parameters, it is best to train through a smaller number of epochs - 5.

Further experiments reference to training of a small number of parameters. Firstly, we trained only 0.42 percent of additional parameters, which was achieved by setting parameter r in LoraConfig to 16. Although only 0.42 percent of the parameters are trained in 5 epochs, we can see that the accuracy has decreased by only two percent compared to the best accuracy obtained when all parameters were trained. Moreover, the time needed for training has decreased significantly compared to the same number of epochs when all parameters were trained. With the increase in the number of epochs to 10, the accuracy improved slightly and with 20 epochs accuracy improved to 89 percent. Although the accuracy improved by 3 percent, the time needed for fine-tuning increased 4 times. The question arises whether such a small improvement in results is worth the extra time required for training. Compared to the previous approach, where all parameters were trained, the memory consumption was reduced almost twice – only 5845MiB were needed for fine-tuning.

The next step in experiments included training about 1.05 of model parameters which was achieved by setting parameter r in LoraConfig to 64. Compared to previous experiments, the training time has slightly increased, but not significantly. In terms of memory consumption and model accuracy, there are no significant differences. The final step in experiments with the English language included setting parameter r in LoraConfig to 128 which resulted in a number of trainable parameters being 1.87%. However, there are no significant differences in time and memory consumption, and model performance as well. We can see that even if we don't train all model parameters, we can achieve results that are on par with the results obtained when all parameters are trained. As well, time and memory consumption are reduced.

Table II shows the results of experiments with text in the Serbian language. When all parameters are trained, a good accuracy of about 85 percent is obtained with 5 epochs of training. It can be observed that with an increase in the number of epochs, model performance decreases slightly, as with the text in the English language.

When the number of trainable parameters is reduced to 0.42% and fine-tuning is conducted in 5 epochs, the model predictions get worse significantly, because accuracy decreases to 75 percent. However, a slight increase in the number of epochs to 10 improved model accuracy to 84 percent, while the time needed for fine-tuning remains significantly less compared to the smallest time obtained when all parameters are trained. When the number of trainable parameters is increased to 1.05%, memory and time consumption slightly increased, but overall model performance did not change significantly for all tested number of epochs. When the number of trainable parameters was about 1.87%, the model achieved better performance for 5 epochs of training with an accuracy of 79 percent. However, model performance did not change significantly for 10 and 20 epochs compared to previous PEFT experiments in the Serbian language.

Comparing results obtained for the English and the Serbian language, it can be observed that in all cases of model training, model performance on English reviews is slightly better. This

TABLE II. RESULSTS OF EXPERIMENTS IN THE SERBIAN LANGUAGE

Trainable %	Time	Memory	Epochs	Accuracy
100	4m 13s	6439MiB	5	0.8571
100	8m 41s	6439MiB	10	0.8432
100	17m 50s	6439MiB	20	0.8397
0.42329481296123295	1m 13s	2017MiB	5	0.7456
0.42329481296123295	2m 27s	2017MiB	10	0.8432
0.42329481296123295	4m 55s	2017MiB	20	0.850174 216
1.050342201278925	1m 15s	2047MiB	5	0.7561
1.050342201278925	2m 31s	2047MiB	10	0.8397
1.050342201278925	5m	2047MiB	20	0.8467
1.8742231413082096	1m 17s	2069MiB	5	0.7944
1.8742231413082096	2m 35s	2069MiB	10	0.8467
1.8742231413082096	5m 11s	2069MiB	20	0.8397

can be expected because, although the model is multilingual, the Serbian language is low-resourced and the quality of data sets may vary. Taking into account the time and memory required for English and Serbian language, we can see that training in the English language takes much time and memory. Although the number of instances in the dataset is the same, the reviews in the English dataset are significantly longer and this leads to more resource consumption. It can be observed that when PEFT is used for a smaller number of epochs, there is a difference between the Serbian and English languages. The model performance for the Serbian language drops significantly.

IV. CONCLUSION

In the paper, we compared traditional and parameter-efficient fine-tuning of large language models on sentiment analysis tasks in the English and Serbian languages. Experiments have shown that we can get equally good results if we train the model with the PEFT approach, whereby the time and memory required for model training are significantly reduced.

All experiments on the English dataset gave good results with an accuracy of 85 percent or higher. Looking at the results for the English language, the best solution is the experiment in which 0.42% parameters of the model were trained through 5 epochs and where an accuracy of 86 percent was achieved. Although the accuracy of the model is 3 percent lower than the best results obtained for English in this paper, this approach is preferred due to the significant reduction in time and memory.

The range of accuracy obtained in the experiments for the Serbian language varies from 75 to 85 percent and is higher than the range for the English language. In the Serbian language, worse results were obtained for the smaller number of epochs. Looking at the results for the Serbian language, the best solution is the experiment in which 0.42% parameters of the model were trained through 10 epochs and where an accuracy of 84 percent was achieved. In this experiment, the time required for training was reduced almost twice and the memory almost three times. It can be observed that with a multilingual model like XLM-RoBERTa, we can achieve results that are on par with results obtained in the experiments with reviews in the English language.

In this paper, it is proved that we can achieve good model performance with non-traditional fine-tuning methods like PEFT. Results obtained with parameter-efficient fine-tuning are almost the same as those obtained with traditional fine-tuning. As well, time and computational resources needed for training are saved. The further direction of this work can include the use of quantization techniques to reduce the size of the model.

ACKNOWLEDGMENT

This work was supported by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia [grant number 451-03-65/2024-03/200102].

REFERENCES

[1] N. Houlsby, A. Giurgiu, S. Jastrzębski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, S. Gelly, "Parameter-Efficient Transfer Learning for NLP," in Proceedings of the 36 th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

[2] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, i V. Stoyanov, "Unsupervised Cross-lingual Representation Learning at Scale," in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 8440-8451, 2020.

[3] V. Batanović, M. Cvetanović, B. Nikolić, "A versatile framework for resource-limited sentiment articulation, annotation, and analysis of short texts" PLoS ONE, vol. 15, no. 11, 2020.

[4] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, i C. Potts, "Learning Word Vectors for Sentiment Analysis," in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, Oregon, USA, pp. 142-150, 2011.

[5] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, i W. Chen, "LoRA: Low-Rank Adaptation of Large Language Models," in Proceedings of International Conference on Learning Representations, 2022.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, i I. Polosukhin, "Attention is All You Need," in Proceedings of 31st Conference on Neural Information Processing Systems (NIPS 2017), pp. 5998-6008, Long Beach, California, USA, 2017.

[7] J. Devlin, M. W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirec-tional Transformers for Language Understanding," in Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), pp. 4171-4186, Minneapolis, Minnesota, Unit-ed States, 2019.

[8] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, "Improving language under-standing by generative pre-training," unpublished.

[9] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, i V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," unpublished.

[10] L. Xu, H. Xie, S.-Z. J. Qin, X. Tao, i F. L. Wang, "Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment," unpublished.

[11] Hugging face, "Transformers," huggingface.co, https://huggingface.co/docs/transformers/index, (accessed April. 18, 2024).

[12] Hugging face, "Peft," huggingface.co, https://huggingface.co/docs/peft/index, (accessed April. 18, 2024).