Записки научных
семинаров ПОМИ
Том 540, 2024 г.

N. Sukhanovskii, M. Ryndin

MMA: A FIGHT FOR MULTILINGUAL MODELS ACCELERATION

Abstract. In this work we focus on common NLP model design: fine-tuning a multilingual language model with data for the target task in one language to solve this task in a different target language. We aim to determine how popular speedup techniques affect multilingual capabilities of Transformer-based model and additionally research the usage of this techniques in combination. As a result, we obtain the NERC model that can be effectively inferred on CPU and keeps multilingual properties across several test languages after being tuned and accelerated with only English data available.

§1. Introduction

Multilingual named entity recognition (NER) tasks are particularly important in facilitating cross-lingual information retrieval, entity-based sentiment analysis, and multilingual document classification. By accurately identifying and categorizing named entities across different languages, multilingual NER systems enable organizations to extract actionable insights from diverse textual data sources, regardless of language barriers. However, the computational demands of Transformer-based models pose challenges, particularly in resource-constrained environments. This work focuses on investigating and implementing methods to accelerate neural networks, particularly models based on transformer architecture. The aim is to enhance the efficiency of these models while retaining their multilingual capabilities, specifically in the context of NER. Also it is worth mentioning that datasets for many classes are often not available in many languages and it is common to use solutions based on multilingual models. This backbone is then finetuned for a target task on single-language data and used on target language data. This approach is working thanks to the backbone's multilingual capabilities, and it is unclear how strong is multilingual generalization after applying acceleration methods that "see" only single-language data. By comparing selected algorithms based on their impact on model quality and speedup achieved on CPU execution, this study aims to identify

Key words and phrases: BERT, pruning, quantization, NERC.

an optimal combination of techniques. To address this, techniques such as quantization, pruning, and model distillation are explored, among others, to reduce model complexity and enhance inference speed on a CPU. Evaluation criteria include assessing the loss in model quality on standard NER datasets and measuring the achieved speedup during CPU execution. The main novelty of this work lies in the measurement of retention of multilingual knowledge affected by techniques which reduce model size. The ultimate goal is to identify a methodology that allows the model to maintain its multilingual capabilities while significantly improving its efficiency on a CPU. Despite our focus on performance on the CPU, all of the techniques discussed will provide speedups for models that run on a GPU if a proper runtime is used. For example, non-structural pruning requires runtime with support for operations with sparse matrices.

§2. Background information

There exist several popular methods to modify models to speed up their inference; the most widely used methods include pruning, distillation, and quantization.

2.1. Pruning. Pruning entails the removal of redundant model components, which can be categorized into structural pruning, involving the deletion of model chunks, and unstructured pruning, which involves the deletion of individual weights in tensors of the model. For BERT-based models, structural pruning commonly targets attention heads, which can be iteratively removed [3]. Conversely, unstructured pruning techniques vary based on criteria for selecting weights for deletion. The metric that determines the importance of heads for deletion is based on how deletion of each individual head will affect the loss function.

The main difference between methods of unstructured pruning lies in how weights are chosen for deletion. The first intuitive approach is to delete weights with lowest absolute values iteratively, while the model is trained or fine-tuned [23]. The main downside of this approach is that when it is used with a pre-trained model some of the weights are not applicable to fine-tuning tasks. To address this issue, a new metric of the "importance" of weights was proposed in the work [6]. The main idea is to remove weights with the lowest change during fine-tuning on target tasks, hence its name is *movement pruning.* The current state of the art method of unstructured pruning for Transformer-based networks is the

Optimal Brain Surgeon pruning approach [9,15]. This method uses second-order information to determine weight deletions that minimize adverse effects on model performance. The main downside of this method is its high computational cost.

2.2. Quantization. Quantization, another approach to accelerate model inference, involves converting high-precision numerical variables, such as real numbers (FP32), into lower-precision variables, such as integers (INT8). Notably, quantization techniques can be further categorized into static, dynamic, and Quantization-Aware Training (QAT), each offering distinct advantages and trade-offs in terms of inference speed and model accuracy [24]. In case of static quantization, the quantization levels are determined using sample data and remain consistent after conversion. Dynamic quantization, on the other hand, adjusts the quantization levels based on the input data characteristics during inference of the model. Quantization-Aware Training (QAT) is a method employed during model training where the model is trained while considering the effects of quantization, allowing it to adapt its parameters to better accommodate the quantization process. This involves simulating the effects of quantization during training, enabling the model to learn more robust representations that preserve accuracy when deployed with reduced precision.

2.3. Distillation. Distillation of neural networks is the process of training a lightweight model called the student from the knowledge of a more complex model, called the teacher. The most common algorithm is to add a term to the student's loss function that is responsible for how much the student's output coincides with that of the teacher. As a result, the student model, in addition to regular learning, also learns to mimic the distribution of teacher responses [5]. Architectures of student models can vary [12], and the loss of the student model can include terms that correspond to similarity weight tensors between the student and the teacher [16, 20].

§3. Experimental Setup

We have conducted our experiments on two datasets: conll2002 [17] and conll2003 [21], commonly used to evaluate models on named entity recognition tasks. The data consists of sentences in 3 different languages: English (en), Dutch (nl), and Spanish (es), with annotations for persons, organizations, locations and other classes. We used only the training split of conll2003 (English language sentences) for fine-tuning and acceleration

Table 1. Results of baseline models.

Model	es dev	es test	en dev	en test	nl dev	nl test
mlbert	0.747	0.749	0.958	0.910	0.806	0.782
xlm	0.777	0.776	0.954	0.922	0.809	0.785

Table 2. Throughput result of baseline models.

Model	Throughput	Batch size	Runtime
Base model	27.9	4	deepsparse
	18.7	4	onnxruntime

procedures, so the model has never seen any sentences for the target languages it is evaluated on (Spanish and Dutch). Also, to simplify the understanding of results we introduced a new metric, the *multilingual score* (mls). Mls is calculated as the sum of differences in the F1-score between base and target models on all non-English datasets. For performance testing, all models were converted into ONNX format and were tested in different inference engines (ONNX runtime[1], deepsparse engine[2]) and with different batch sizes (1, 4, 8) and fixed sequence length of 128 tokens. The main metric for the models is the F_1 score on each subset and metric for performance is throughput on CPU (i7-11800H). Baseline models are pre-trained BERT-base-multilingual-cased(mlbert) [19] and XLM-RoBERTa-base(xlm) [18] from Hugging Face[3] hub. These models were chosen because they are the most popular BERT-like multilingual models. Their fine-tuned or pre-trained versions were used as starting point for all experiments. Their baseline results are shown on table 1 and 2.

3.1. Structural pruning.

3.1.1. *Are sixteen heads really better than one? [3].* The initial point of our approach involved fine-tuning the Multilingual BERT model. During each iteration of the algorithm, we selected the 12 attention heads with the lowest metric. Masked heads were excluded from these calculations.

[1]https://github.com/onnx/onnx
[2]https://github.com/neuralmagic/deepsparse
[3]https://huggingface.co/

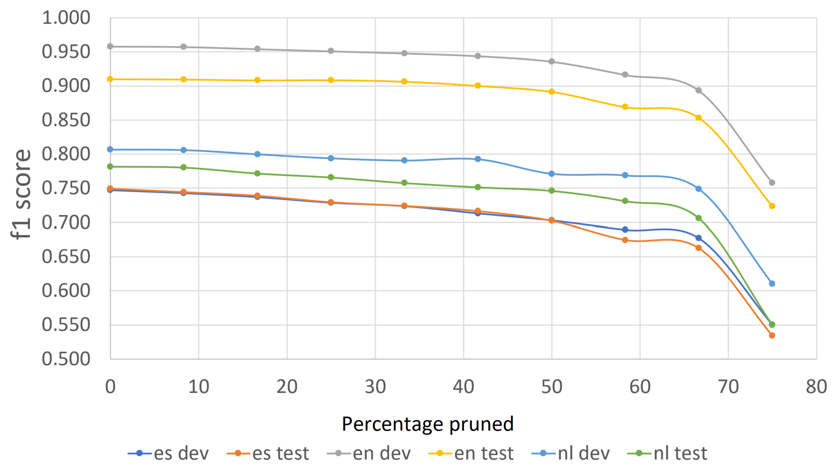

Figure 1. Results of models with different numbers of attention heads.

Subsequently, another set of 12 attention heads was chosen from the remaining heads, ensuring that at least one unmasked head remained on each layer. Upon conversion to a computational graph, the masked attention heads were completely removed. Figure 1 illustrates that a reduction of approximately 50% in the number of removed heads corresponds to a decline in the F1 score across all languages. This decline closely aligns with findings reported in the original article for similar tasks. Notably, despite the assessment of attention head “importance” being conducted solely on the English subset, the reduction in F1 score remains consistent across all languages.

3.1.2. *TextPruner [1].* The same model utilized in the previous experiment served as the foundation for this one. As before, 12 heads were masked at each iteration; however, in this instance, a configuration was implemented where the number of masked heads on each layer remained consistent. The reduction in the hidden size of the feed forward layer at each step mirrored that of the previous experiment, resulting in a diminishing discrepancy in layer sizes across iterations. All other parameters were maintained at similar levels to those of the prior experiment. Figure 2 depicts the quality of the model throughout iterative head removal. Notably, due to the fact

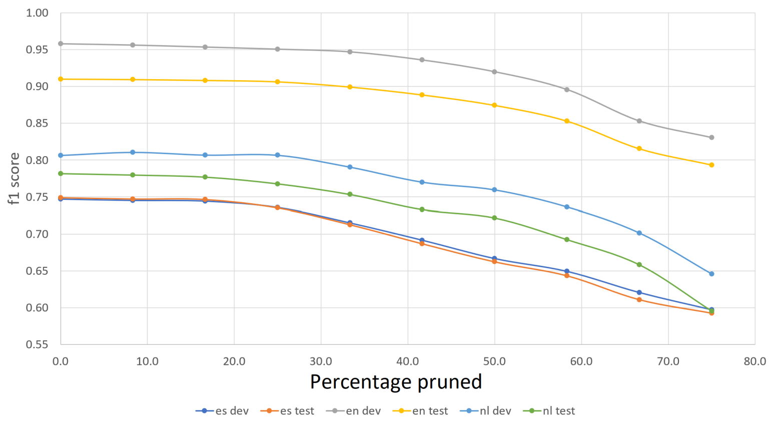

Figure 2. F_1 score (TextPruner).

that the number of heads on each layer should remain constant, the overall quality of the model was adversely affected. This decline in performance begins not at a removal rate of 50%, as observed in previous instances, but rather at 30%. This observation suggests that the significance of heads on individual layers varies, and the constraint of uniform removal impedes the model's performance. Furthermore, despite the presumption that an equal distribution of heads across layers would enhance model performance, the conducted experiments did not substantiate this hypothesis. Model inference result for structural pruning can be found in Appendix A.

3.2. Unstructured pruning. The base models utilized in this case were XLM-RoBERTa-base and BERT-base-multilingual-cased. The fine-tuning process consisted of three stages: 1) initial tuning for the task, 2) iterative removal of weights, and 3) final additional training of an already sparse model. Similar to previous experiments, weights were iteratively masked based on a specified metric, accounting for previously masked weights in the calculations. Upon completion of training, the model was exported to the ONNX format in a specialized manner, enabling the exploitation of the resulting matrices' sparsity to accelerate model inference. Throughout training, a cyclic scheduler was employed, following recommendations outlined in the referenced article [10]. Notably, this scheduler proved particularly effective for pruning, as a comparatively high learning rate during

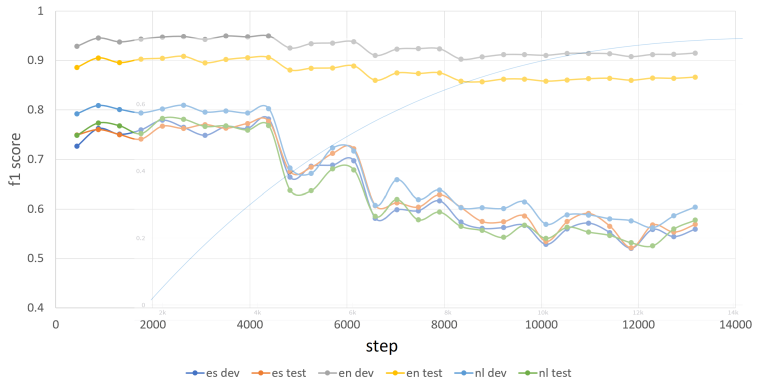

Figure 3. F_1 score on various languages during pruning with the OBS algorithm up to 90% sparsity for the xlm-roberta model with a cubic curve of sparsity percentage.

certain phases facilitated model adaptation to weight removal, while subsequent reductions in the learning rate stabilized model performance. The efficacy of this scheduler was validated through both personal experimentation and the findings presented in the article [11].

Furthermore, each pruning algorithm featured specific parameters, initially drawn from their respective original articles and subsequently refined where computational resources permitted. However, a comprehensive search for hyperparameters for each algorithm was not undertaken. The mask update rate served as a common parameter across all considered pruning algorithms and remained consistent throughout the experiments.

The models underwent training to achieve 80% and 90% sparsity of the weight matrices, serving as a global parameter for the entire model. Hence, while individual structural elements of the model could exhibit varying degrees of sparsity, the cumulative sparsity equated to the specified percentage. The selected sparsity levels were chosen to balance noticeable acceleration while maintaining robust performance across datasets.

During each training iteration, the model pursued a target sparsity percentage, and the manner in which this transition occurred from 0 to the specified percentage influenced the final model quality. Two key parameters influenced this transition: the initial sparsity, denoting the significant

Table 3. Results of models with different pruning configurations.

Model	**es dev**	**es test**	**en dev**	**en test**	**nl dev**	**nl test**	**mls dev**	**mls test**
mlbert magnitude 80	0.128	0.076	0.892	0.790	0.068	0.057	-1.357	-1.398
mlbert movement 80	0.117	0.068	0.897	0.783	0.066	0.055	-1.37	-1.408
mlbert obs 80	0.648	0.646	0.935	0.888	0.622	0.589	-0.283	-0.296
xlm magnitude 80	0.483	0.484	0.901	0.855	0.524	0.490	-0.546	-0.557
xlm movement 80	0.481	0.485	0.903	0.851	0.527	0.491	-0.545	-0.555
xlm obs 80	0.628	0.641	0.933	0.882	0.651	0.604	-0.274	-0.286
xlm magnitude 90	0.533	0.536	0.904	0.859	0.498	0.464	-0.522	-0.531

percentage of weights removed at the onset of pruning, and the curve dictating the trajectory towards the target sparsity percentage.

In Figure 3 and Figure 4 in Appendix A, the F_1 score across different model languages during pruning is depicted with linear and cubic curves of the target sparsity percentage, respectively. Remaining parameters were held constant. It is evident that the model with a linear curve exhibits notably lower F_1 scores, particularly in languages for which it was not explicitly trained. This decline occurs sharply towards the end of training, within the final 30% of sparsity. Prior to this point, the model with a linear curve outperformed its cubic counterpart. This observation suggests that the initial 40% of sparsity has minimal impact on model quality across all languages, but as sparsity increases, the decline in quality becomes more significant. Notably, this dependency is nonlinear, mirroring the resulting acceleration of the model.

Table 3 presents the overall results. The models' names consist of three parts: 1) architecture (mlbert — Multilingual BERT, xlm — Multilingual RoBERTa), 2) pruning algorithm (magnitude — based on absolute values, movement — based on gradients, OBS — based on second-order information, the Hessian), and 3) percentage of model sparsity.

Key takeaways from these results include the observation that higher-order pruning algorithms generally yield better performance. However, algorithms based on absolute values and gradients incur minimal additional computational load beyond regular training, whereas OBS not only prolongs training time by at least threefold but also demands substantial video memory, necessitating a professional graphics card for optimal operation.

Table 4. Results of distilled models in various languages.

Model	**es dev**	**es test**	**en dev**	**en test**	**nl dev**	**nl test**	**mls dev**	**mls test**
DistillBERT with teacher	0.708	0.729	0.947	0.900	0.802	0.758	-0.04	-0.04
DistillBERT without teacher	0.690	0.699	0.942	0.901	0.760	0.725	-0.10	-0.11
MiniLM	0.763	0.757	0.945	0.905	0.793	0.761	-0.02	-0.01

In addition, it is important to note that first- and second-order algorithms substantially diminish the multilingual capabilities of the BERT model, unlike RoBERTa. Models pruned using second-order information-based algorithms retain 95% of the base model's F1 score in English and 81% to 85% in other languages.

3.3. Distillation. The base model was distilbert-base-multilingual-cased, which was trained on Wikipedia articles in 104 of the most common languages using bert-base-multilingual-cased as a teacher. The basic model was fine-tuned with and without a teacher already on the task of recognizing nominal entities. A finely tuned xlm-roberta-base was used as the teacher and only the label probability distribution from the teacher was used for the student error function. Also for comparison, we took a MiniLM model pre-trained in several languages, which was fine-tuned without a teacher on an English-language data set. The choice of distilled models is limited, because there is almost none models whose were pretrained on multilingual data.

Table 4 presents the results of distilled models in various languages. For the Supervised DistilBERT model, it retains 98.5% of the F_1 score compared to basic BERT in English and 95% to 97% in other languages. The addition of a teacher model during fine-tuning does not alter the score in English but improves it by 5% in other languages. This marginal improvement in utilizing a teacher can be attributed to DistilBERT already having a teacher during pre-training.

MiniLM, on the other hand, shows a minor decrease compared to basic BERT, trailing by only 1% in English and 2% in Dutch, while outperforming it by 4% in Spanish. The discrepancy in Spanish performance can be

Table 5. Throughput result of distilled models.

Model	**Throughput**	**Batch size**	**Runtime**
DistillBERT	45.3	1	deepsparse
	50.3	4	deepsparse
	46.6	8	deepsparse
	34.6	1	onnx
	34.3	4	onnx
	31.3	8	onnx
miniLM	93.8	1	deepsparse
	111.8	4	deepsparse
	112.7	8	deepsparse
	64.9	1	onnx
	70.0	4	onnx
	60.1	8	onnx

attributed to variations in the dataset used for pre-training and distillation.

Overall, distillation proves to be an effective method, not only replicating the results of the basic model but also preserving the multilingual nature of the model. It is noteworthy that the selected models were distilled during model pre-training, while other methods were employed only during fine-tuning. Additionally, it is important to highlight the computational cost associated with training, as both teacher and student models need to be maintained in memory and executed, particularly for complex training processes like MiniLM.

Table 5 presents the performance results of the distilled models. Since all models diverge from baseline ones at the architectural level, the performance gains are visible in runtimes. Since the Distillbert architecture is basically a BERT with half as many layers, the performance gain is almost 90 percent. As for miniLM, the main contribution to the speedup of the model is made by reducing the hidden size of weights by 2x, which, according to the original work [22], gives a speedup of 2.7x. In our experiment, the speed of the model increased by 3.8x compared to the baseline BERT. This difference is due to the fact that testing is carried out in inference engines and they are presumably better optimized for working with smaller matrix sizes.

Table 6. Results of quantized models.

Quantization type	**es dev**	**es test**	**en dev**	**en test**	**nl dev**	**nl test**	**mls dev**	**mls test**
Dynamic	0.748	0.761	0.946	0.899	0.792	0.778	-0.013	-0.014
Static	0.74	0.754	0.935	0.891	0.784	0.772	-0.029	-0.027
QAT	0.752	0.767	0.948	0.904	0.795	0.781	-0.006	-0.005
Without data	0.733	0.748	0.924	0.881	0.775	0.761	-0.045	-0.044

3.4. Quantization. A fine-tuned BERT-base-multilingual-cased model served as the starting point for quantization, consistent with previous experiments. The calibration dataset for static quantization comprised the training portion of the CoNLL 2003 dataset. Dynamic quantization was implemented using ONNX, thereby "hard-wiring" it into the model file and ensuring compatibility with all launch environments that store models in the ONNX format.

During quantization-aware training, the model underwent fine-tuning for the same number of epochs as in the base model fine-tuning process. To establish quantization boundaries in scenarios where data was absent, randomly generated and nonsensical input data was utilized.

Table 6 presents the impact of different quantization methods on the final quality and multilinguality of the model.

The quantization method during training yields the highest quality results, as expected, since it requires a substantial amount of real data and computational resources for implementation. Dynamic quantization is the optimal method for maintaining model quality without additional data, while static quantization, despite the presence of a calibration dataset, still yields slightly lower quality compared to dynamic quantization. Dataless quantization provides the fastest model generation but sacrifices quality.

Interestingly, the results of models in non-target languages decline proportionally to those in English, indicating that quantization does not negatively impact the multilinguality of the model. Overall, the additional computational costs associated with implementing quantization during training are deemed insignificant, making this method preferable, especially when access to training data is available. Moreover, a model quantized during training retains over 98% of the metrics of the base model, emphasizing its effectiveness.

Table 7. Throughput result of quantized models.

Quantization type	**Throughput**	**Batch size**	**Runtime**
Dynamic	76,3	8	deepsparse
Static	96,5	8	deepsparse

Table 7 presents the performance results of various quantization approaches. The full table can be found in Appendix A. The table shows only two methods, since the difference between the methods lies in the method for converting the weights, and since all the final weights are determined before running the model in case of static quantization, there is no difference in performance. For static methods, the speedup is 340 percent. Since dynamic quantization incurs additional computational overhead during model execution, the speedup is only 270 percent.

3.5. Combined approach. To streamline the presentation, we will only showcase the best configurations identified from previous chapters in table 8. For instance, the pruning method based on second-order information demonstrated superior performance in preserving both multilinguality and accuracy across languages. All types of quantization were found to be equally compatible with the described algorithms and can be applied to fully trained models. However, quantization during training was found to best preserve model accuracy, making it the preferred choice for combination.

During non-structural pruning of models, fine-tuning was conducted using the same approach and model as described in the chapter on distillation, including the utilization of teacher models for guidance. This consistent approach ensures comparability and reliability across experiments. The main outcome is that techniques combination does not create any special properties: resulting model has aggregated speedup and trade-offs of corresponding techniques but nothing extra. Full table with results of throughput can be found in Appendix A.

§4. Conclusion

Throughout this study, we have explored key techniques for reducing and accelerating neural network models, including quantization, pruning,

Table 8. Results of best combined methods.

Model	**es dev**	**es test**	**en dev**	**en test**	**nl dev**	**nl test**	**mls dev**	**mls test**
xlm 80 q	0.610	0.623	0.906	0.857	0.632	0.587	-0.31	-0.32
xlm 90 q	0.587	0.600	0.873	0.825	0.609	0.565	-0.36	-0.37
distillbert q	0.687	0.708	0.9	0.9	0.779	0.736	-0.09	-0.09

Table 9. Throughput result of final models.

Model	**Throughput**	**Batch size**	**Runtime**
xlm 80 q	124.87	4	deepsparse
xlm 90 q	219.44	4	deepsparse
distillbert q	158.44	8	deepsparse

distillation, and their application to Transformer-based models in multilingual named entity recognition tasks. For pruning, sparsity thresholds that preserve multilingual capabilities were identified. Research revealed varying support for sparse matrix operations across different runtimes, with the deepsparse framework proving advantageous for sparse matrix models. Notably, the second-order pruning method (OBS) emerged as the most effective, albeit demanding increased training equipment resources.

Among the quantization methods examined, all yielded modest speed enhancements with negligible impact on the models' multilingual nature. These methods have broad software support and entail manageable computational overhead. Our experiments recommend the use of static quantization with boundaries determined during model training (QAT).

Distilled models surpassed their full-scale counterparts while maintaining quality metrics.

Combining the most promising acceleration methods yielded compelling results. Models produced using proposed approach demonstrated a 1.5 to 2 times speed improvement over individually applied methods and an 11.5-fold enhancement over baseline models, all while preserving progressive multilingual quality metrics.

§5. Additional figures and tables

In the appendix, we show additional figures and tables with experimental results from our study.

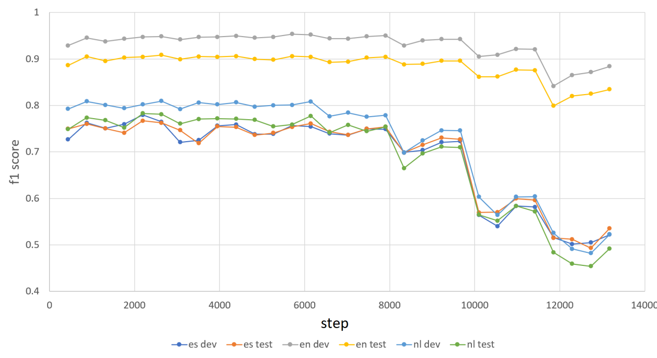

Figure 4. F_1 score on various languages during pruning with the OBS algorithm up to 90% sparsity of xlm-roberta model with a linear curve of sparsity percentage.

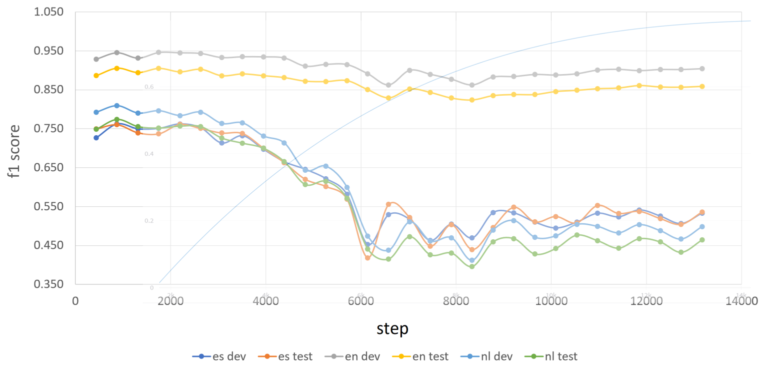

Figure 5. F_1 score on various languages during pruning with the magnitude pruning algorithm up to 90% sparsity of xlm-roberta model with a cubic curve of sparsity percentage.

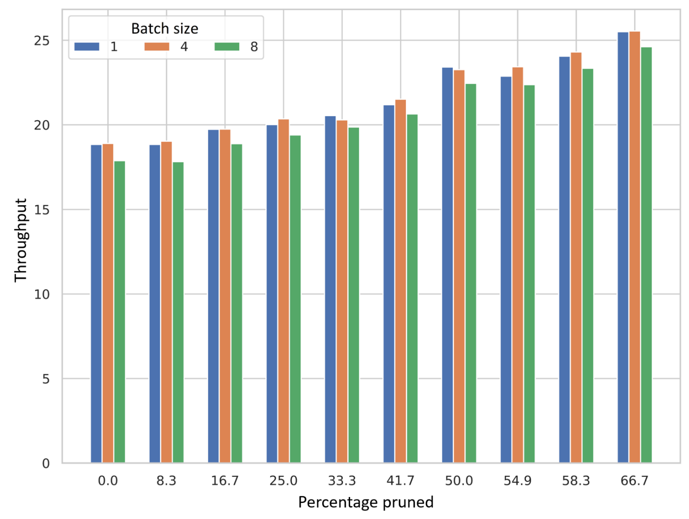

Figure 6. Throughput results for Are sixteen heads really better than one? in onnxrunime.

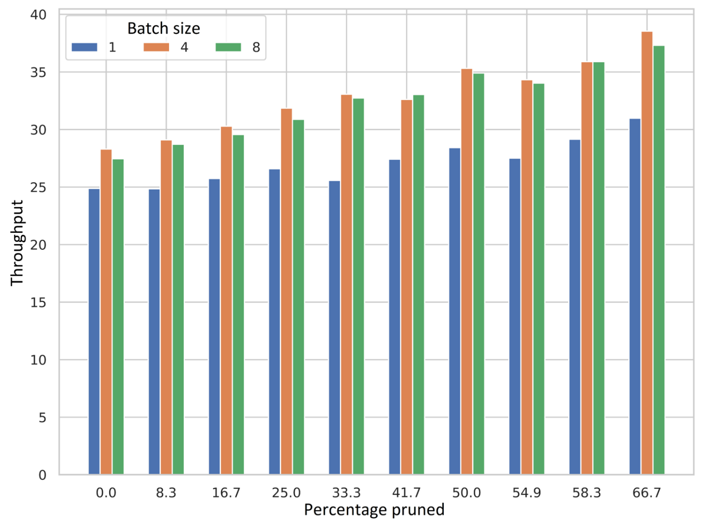

Figure 7. Throughput results for Are sixteen heads really better than one? in deepsparce engine.

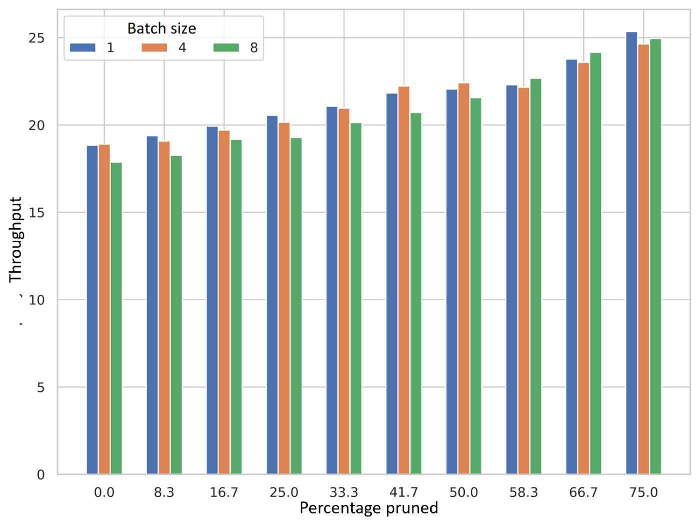

Figure 8. Throughput results for TextPruner in onnxrunime.

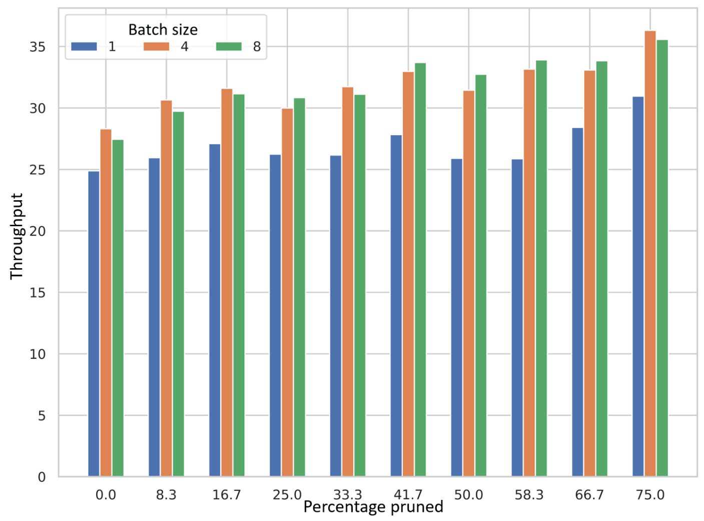

Figure 9. Throughput results for TextPruner in deepsparce engine.

Table 10. Throughput result of final models.

Model	**Throughput**	**Batch size**	**Runtime**
xlm 80 q	85.32	1	deepsparse
	124.87	4	deepsparse
	113.40	8	deepsparse
	38.75	1	onnx
	35.33	4	onnx
	30.04	8	onnx
xlm 90 q	162.75	1	deepsparse
	219.44	4	deepsparse
	199.97	8	deepsparse
	38.09	1	onnx
	34.46	4	onnx
	29.94	8	onnx
distillbert q	133.34	1	deepsparse
	154.27	4	deepsparse
	158.44	8	deepsparse
	71.91	1	onnx
	56.98	4	onnx
	54.87	8	onnx

Table 11. Throughput result of quantized models.

Quantization type	**Throughput**	**Batch size**	**Runtime**
Dynamic	53,6	1	deepsparse
	74,6	4	deepsparse
	76,3	8	deepsparse
	30,7	1	onnx
	28,1	4	onnx
	20,9	8	onnx
Static	67,8	1	deepsparse
	94,5	4	deepsparse
	96,5	8	deepsparse
	38,9	1	onnx
	35,5	4	onnx
	26,5	8	onnx

References

1. Z. Yang, Y. Cui, and Z. Chen, *TextPruner: A Model Pruning Toolkit for Pre-Trained Language Models*, Proc. 60th Annual Meet. Assoc. Comput. Linguist.: Syst. Demonstr. (2022), pp. 35–43.

2. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, ArXiv preprint arXiv:1810.04805 (2019).
3. P. Michel, O. Levy, and G. Neubig, *Are Sixteen Heads Really Better than One?.* — Adv. Neural Inf. Process. Syst. **32** (2019).
4. E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, *Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned*, Proc. 57th Annual Meet. Assoc. Comput. Linguist. (2019), pp. 5797–5808.
5. V. Sanh, L. Debut, J. Chaumond, and T. Wolf, *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*, ArXiv preprint arXiv:1910.01108 (2019).
6. F. Lagunas, E. Charlaix, V. Sanh, and A.M. Rush, *Block Pruning For Faster Transformers*, ArXiv preprint arXiv:2109.04838 (2021).
7. J.S. McCarley, *Pruning a BERT-based Question Answering Model*, ArXiv preprint arXiv:1910.06360 (2019).
8. T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, *Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks.* — J. Mach. Learn. Res. **22**(1) (2021), pp. 10882–11005.
9. E. Kurtic, D. Campos, T. Nguyen, E. Frantar, M. Kurtz, B. Fineran, M. Goin, and D. Alistarh, *The Optimal BERT Surgeon: Scalable and Accurate Second-Order Pruning for Large Language Models*, ArXiv preprint arXiv:2203.07259 (2022).
10. L.N. Smith, *No More Pesky Learning Rate Guessing Games*, ArXiv preprint arXiv:1506.01186 (2015).
11. E. Kurtic, D.F. Campos, T. Nguyen, E. Frantar, M. Kurtz, B. Fineran, M. Goin, and D. Alistarh, *The Optimal BERT Surgeon: Scalable and Accurate Second-Order Pruning for Large Language Models.* — Conf. Empir. Methods Nat. Lang. Process. (2022).
12. W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers*, ArXiv preprint arXiv:2002.10957 (2020).
13. Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, *RoBERTa: A Robustly Optimized BERT Pretraining Approach*, ArXiv preprint arXiv:1907.11692 (2019).
14. S.P. Singh and D. Alistarh, *WoodFisher: Efficient Second-Order Approximation for Neural Network Compression*, ArXiv preprint arXiv:2004.14340 (2020).
15. B. Hassibi and D.G. Stork, *Second Order Derivatives for Network Pruning: Optimal Brain Surgeon.* — Adv. Neural Inf. Process. Syst. (1992).
16. X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, *TinyBERT: Distilling BERT for Natural Language Understanding*, ArXiv preprint arXiv:1909.10351 (2020).
17. E.F. Tjong Kim Sang, *Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition*, ArXiv preprint arXiv:cs/0209010 (2002).
18. A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, *Unsupervised Cross-lingual Representation Learning at Scale*, ArXiv preprint arXiv:1911.02116 (2020).

19. T. Pires, E. Schlinger, and D. Garrette, *How multilingual is Multilingual BERT?*, ArXiv preprint arXiv:1906.01502 (2019).
20. Z. Yang, Y. Cui, Z. Chen, W. Che, T. Liu, S. Wang, and G. Hu, *TextBrewer: An Open-Source Knowledge Distillation Toolkit for Natural Language Processing*, Proc. 58th Annual Meet. Assoc. Comput. Linguist.: Syst. Demonstr. (2020), pp. 9–16.
21. E.F. Tjong Kim Sang and F. De Meulder, *Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition*, ArXiv preprint arXiv:cs/0306050 (2003).
22. W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers*, ArXiv preprint arXiv:2002.10957 (2020).
23. T. Gale, E. Elsen, and S. Hooker, *The State of Sparsity in Deep Neural Networks*, ArXiv preprint arXiv:1902.09574 (2019).
24. H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, *Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation*, ArXiv preprint arXiv:2004.09602 (2020).

Ivannikov Institute for System Programming
of the Russian Academy of Sciences, Moscow, Russia
E-mail: {sukhanovskii.nl, mxrynd}@ispras.ru

Поступило 15 ноября 2024 г.