MOROCCO: Model Resource Comparison Framework

Valentin Malykh[1], Alexander Kukushkin[2], Maria Tikhonova[3], and Tatiana Shavrina[3]

[1] MISiS University valentin.malykh@phystech.edu
[2] Alexander Kukushkin Data Science Laboratory alex@alexkuk.ru
[3] HSE University m_tikhonova94@mail.ru, rybolos@gmail.com

Abstract. A new generation of pre-trained transformer language models has established new state-of-the-art results on many tasks, even exceeding the human level in standard NLU benchmarks. Despite the rapid progress, the benchmark-based evaluation has generally relied on the downstream task performance as a primary metric, limiting the scope of model comparison in their practical use, which is also limited by the resources required by the models to run. This paper presents MOdel ResOurCe COmparison (MOROCCO), a publicly available framework[4] that allows assessing models concerning their downstream quality, combined with two computational efficiency metrics such as memory consumption and throughput during the inference stage. The framework allows flexible integration with popular leaderboards compatible with jiant environment, e.g. SuperGLUE. We demonstrate the MOROCCO applicability by evaluating ten transformer models on two multi-task GLUE-style benchmarks in English and Russian and provide the model analysis.

1 Introduction

The field of NLP has been centered around the "pre-train & fine-tune" paradigm, which involves pre-training a language model (LM) on an extensive text corpus and its further fine-tuning for a downstream task in a supervised fashion. Many transformer LMs Vaswani et al. (2017) fall under this paradigm which has established new state-of-the-art results for the majority of NLP tasks such as text classification Sun et al. (2019), part-of-speech tagging Tsai et al. (2019), machine translation Zhu et al. (2019), and many others. The models have demonstrated various capabilities, ranging from cross-lingual zero-shot transfer Pires et al. (2019) to generating texts that are hard to distinguish from the human-written ones Zellers et al. (2020), and have even outperformed human solvers in standard NLU benchmarks He et al. (2021).

However, the rich diversity of LMs that differ in the number of parameters and the architecture design Liu et al. (2020) has been mainly assessed by the downstream performance as a primary metric on many standard benchmarks such as GLUE Wang et al. (2018), XGLUE Liang et al. (2020), SuperGLUE Wang et al. (2019) and XTREME Hu et al. (2020). Although the benchmarks provide a

[4] The link is removed for anonymity.

standard for direct model comparison, the performance-oriented approach limits the scope of the evaluation methods Ethayarajh and Jurafsky (2020). Motivated by the need of expanding the methodology, various benchmarks and contests have been proposed targeting computational and technical aspects of the models (see Section 2), highlighting the problem of model scaling Rogers (2019).

In line with these works, we introduce MOdel ResOurCe COmparison (MOROCCO), a publicly available framework for model evaluation in terms of their practical use. The contributions of this paper are framed as follows. First, we present a methodology to measure the downstream performance and computational efficiency of the models in a fixed environment. Second, we present a software framework that is adopted to jiant environment Pruksachatkun et al. (2020) that supports over 50 downstream tasks[5], including GLUE-style ones. We demonstrate the MOROCCO applicability by evaluating ten transformer models on two SuperGLUE benchmarks for English and Russian and provide the model analysis. This way of model evaluation provides the researcher with the opportunity of the model comparison from different perspectives, specifically those that meet the user's needs.

2 Related Work

NLP benchmarks The trend for model-agnostic evaluation has been recently set by canonical multi-task NLU benchmarks such as GLUE Wang et al. (2018) and SuperGLUE Wang et al. (2019). Such evaluation method does not consider any computational and technical aspects of the models that differ significantly by the number of parameters and architecture design choices, such as the number of transformer blocks, attention mechanism, pre-training objectives, etc. Besides, the benchmarks do not support the interaction with the user models, limiting the leaderboard results' reproducibility Rogers (2019); Ethayarajh and Jurafsky (2020).

Efficient NLP The trade-off between model performance and computational efficiency has been explored in multiple shared tasks and competitions. The series of Efficient Neural Machine Translation challenges Birch et al. (2018); Hayashi et al. (2019); Heafield et al. (2020) jointly measured the model downstream performance on the task of machine translation and computational efficiency parameters, ranging from memory consumption to size of a Docker image. The organizers selected the Pareto-optimal solutions Aleskerov et al. (2007), i.e. those that require less computational resources when delivering a prominent downstream performance.

The EfficientQA competition Min et al. (2021) aims at creating effective NLP systems for open-domain question answering (ODQA). The submissions are limited by many performance and technical requirements, which stimulate the community to develop optimal ODQA systems that can achieve superior

[5] https://github.com/nyu-mll/jiant/blob/master/guides/tasks/supported_tasks.md

performance while satisfying the technical needs and operating on an optimal amount of retrieval corpora.

The SustaiNLP challenge Wang and Wolf (2020) targets developing efficient yet accurate models. The efficiency is estimated as the power consumed throughout the inference time calculated utilizing experiment impact tracker Henderson et al. (2020). The submitted systems improve total energy consumption over the BERT-base as much as 20×, but the results are around two points lower on average. Although using the same testbed, the MOROCCO framework was developed with the opposite goal in mind. It provides adequate estimates of how many resources consume the models that reach human-level performance. As MOROCCO supports Docker images, it can be easily integrated into any benchmark or probing task, built upon jiant framework.

Dynaboard Ma et al. (2021) is a cloud-based platform on which a submitted model is evaluated according to five different criteria, including task performance, throughput, memory consumption, fairness, and robustness scores. The aggregating Dynascore is designed according to multi-criteria optimization theory to reflect user preferences. Supported tasks include several NLI, QA, sentiment classification, and hate speech detection. The authors use a few pre-existed datasets and several newly collected ones. This forbids their results to be a direct extension to the SuperGLUE-like leaderboards. Another important point is that their methodology is somewhat questionable in fairness and robustness, since it is subjective to the augmentations they use.

3 Evaluation Framework

MOROCCO can be used to rank the benchmark leaderboard models by computational metrics (see Section 3.1). To demonstrate that MOROCCO is compatible with GLUE-style benchmarks, we perform experiments using SuperGLUE tasks for English and Russian (see Section 3.2) over popular transformer-based models (see Section 3.3), which are publicly released as a part of the HuggingFace library Wolf et al. (2019).

Submission details To evaluate a model's performance in MOROCCO on the [Russian]SuperGLUE tasks, a person should prepare their submission as a Docker container and send it to the testbed. The testbed platform runs the submitted Docker container with limited memory, CPU/GPU, and running time. The container is expected to read the texts from the standard input channel and output the answers to the standard output. During the inference, the running time is recorded for the submission scoring. We perform several runs and compute the median values to eliminate the running time and memory footprint dispersion caused by technical reasons. Next, the output from the container is evaluated with task-specific metrics. The results compute the final evaluation score for the whole submission. To ensure the comparability of the collected metrics, we fix the computation hardware. We use Yandex.Cloud[6] virtual instances, where the

[6] https://cloud.yandex.com/

following hardware is guaranteed: 1 × Intel Broadwell CPU, 1 × NVIDIA Tesla V100 GPU. The Docker containers are equipped with Ubuntu 20.04. Following the SuperGLUE infrastructure, our framework is designed to comprise with jiant framework, alongside simple requirements for the evaluation containers built upon other frameworks. It also can be run locally using the released codebase. The details on the running process are provided in Appendix B.

3.1 Metrics

We report the computational efficiency of the tested model utilizing the memory footprint and inference speed.

Memory footprint implicitly allows accounting for the model's size and the number of weights, as there is strong dependency. To measure model GPU RAM usage M, we run a container with a single record as input, calculate the maximum GPU RAM consumption, repeat the procedure five times, and compute a median value.

Inference speed measures directly how much time the model consumes on specific hardware, implicitly estimating its complexity. To measure the inference speed T_p we first compute T_N, which we run a container with N records as input with batch size 32[7] for. We also estimate initialization time T_{init} with running a container with an input of size 1. Inference speed Tp is computed as follows: $Tp = \frac{N}{T_N - T_{\text{init}}}$. In our experiments, we use $N = 2000$, which the user can adjust. We repeat the procedure five times to compute a median value.

Quality Q is an aggregated value computed from task-specific metric values by averaging. The averaging itself is somewhat questionable aggregation method due to quality measure semantic difference as shown in Shavrina and Malykh (2021), nevertheless we strictly follow the methodology presented in SuperGLUE Wang et al. (2019) and RussianSuperGLUE Shavrina et al. (2020) to ease the integration of our framework with the mentioned ones.

Fitness Overall, our evaluation procedure utilizes three different scores, namely the aggregated performance score Q, the inference speed Tp, and the memory footprint M. We propose to take into account these three characteristics of a model and make an integral measure of its "fitness" F that combines quality and computational metrics. There is a plethora of ways to calculate fitness, but we can propose several assumptions based on common sense. Namely, we suppose that memory consumption should be lowered, while the achieved quality and processing speed should be increased. To achieve this we propose a following formula, which follows an idea from van Rijsbergen (1979):

$$F = f_Q(Q) \times \frac{f_{Tp}(Tp)}{f_M(M)}, \tag{1}$$

[7] The batch size of 32 is chosen empirically and utilizes the GPU almost at 100% on the experiment tasks. Note that it can be adjusted to meet the user's needs.

where Q is the aggregated metric-based score for the specific tasks, M is measured in bytes, Tp is measured in records per second (RPS). The f_* stand for some function which is applied to specified parameter.

Basing on our experiments, we propose to set f_Q and f_{Tp} to identity function I, while f_M should be set to logarithmic function. We ground that on the exponential model size increase as shown in Sanh et al. (2019). We result with the following formula used in the next experiments:

$$F = Q \times \frac{Tp}{\log(M)}. \tag{2}$$

The main idea of the proposed metric is to provide a practitioner with simple to use single metric or "rule-of-thumb" in their search for the best model in mythic general case. We understand that there is no general case and also provide all the underlying evaluation details to make their choice more informed. We provide details on the other possible formulations and their limitations in section 5.

3.2 Tasks

We use all 9 tasks from both SuperGLUE and RussianSuperGLUE. For the tasks' description please refer to Appendix A.

3.3 Models

We run the experiments on the following publicly available models that achieved competitive performance on SuperGLUE and Russian SuperGLUE benchmarks. Models for English include monolingual (en_bert_base) and multilingual base BERT (bert-multilingual) Devlin et al. (2019), RoBERTa-base Liu et al. (2019) (en_roberta_base), ALBERT-base Lan et al. (2019) (albert), and GPT-2-large Radford et al. (2019) (en_gpt2). Models for Russian involve multilingual BERT-base (bert-multilingual), 3 variants of ruGPT-3[8] (rugpt3-small, rugpt3-medium, and rugpt3-large), RuBERT-base (rubert) Kuratov and Arkhipov (2019), and Conversational RuBERT-base[9] (rubert-conversational) trained on social media data.

4 Results

Figure 1 demonstrates the results for Russian SuperGLUE (top) and SuperGLUE for English (bottom) based on the received Q, Tp, and M (see Section 3.1). These figures discover the models' relative positions for both languages. For English language GPT-2 seems to be the least optimal model, while RoBERTa is the most optimal one. One could apply Pareto rule to monolingual BERT and ALBERT models and find them roughly equivalent. For Russian, the most optimal model

[8] https://github.com/sberbank-ai/ru-gpts
[9] https://huggingface.co/DeepPavlov/rubert-base-cased-conversational

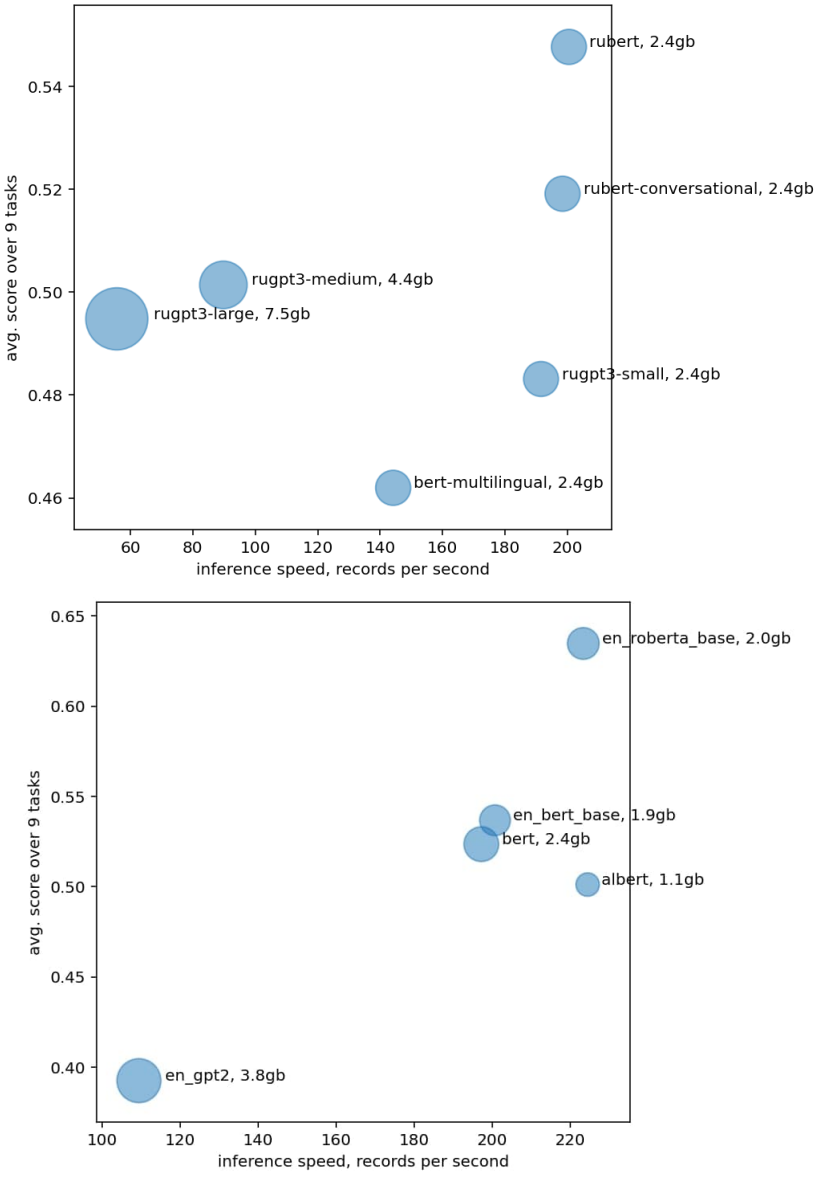

Fig. 1: Model evaluation on RussianSuperGLUE (top) and SuperGLUE (bottom). X-axis=Inference speed Tp (RPS). Y-axis=Task-specific performance Q. The memory footprint M is represented by the size of the circle.

is RuBERT and RuGPT3-large is the least optimal one. Pareto rule gives an equivalence of multilingual BERT and RuGPT3-medium.

The fitness metric F results are presented in Table 1. RoBERTa model had shown the best score for English, while RuBERT is the best fit among the tested models for Russian. The multilingual BERT model showed significantly different results on the two languages. We hypothesize that it attributes to the difference in the datasets in SuperGLUE and RussianSuperGLUE and the model's training data askew towards the English language. Overall, the evaluation results have revealed better models using task-specific quality, memory footprint, and inference speed.

Table 1: Fitness evaluation for the models in English and Russian. The models ordered to ease comparison between the languages.

English		Russian	
en_bert_base	5.05	rubert	4.84
bert-multilingual	4.79	bert-multilingual	3.30
en_roberta_base	6.63	rubert-conversational	4.59
albert	5.41	rugpt3-small	3.89
en_gpt2	1.95	rugpt3-medium	1.89
		rugpt3-large	1.24

5 Discussion

5.1 Detailed Metric Examination

Averaging the estimates of Q, Tp, and M is one of the main limitations of the proposed evaluation procedure. Averaging memory consumption M is less problematic, as it is relatively stable for any reasonable sample size. However, two other metrics require a more detailed investigation. Figure 2 compares the mean and maximum values of Q for different models. Each model was trained five times with different random seeds and was scored ten times, making fifty runs overall. The only exception was made to the largest model, rugpt3-large, fine-tuned only one time. Blue dots present evaluation for a single run, pale red dots show mean results for all runs, and full red dots show the maximum results. The ranking, achieved by maximum and mean scores, is identical.

Figure 3 compares averaged normalized inference speed for different task sets, adopted from RussianSuperGLUE. The normalization is done alongside the X-axis, thus, one can compare the models' ranking for different task sets. The model order remains mostly unchanged, while occasionally, top models exchange positions.

We conclude that our evaluation procedure is stable. Averaging the estimates of Q, Tp, and M does not introduce issues to the evaluation procedure and makes model comparison informative.

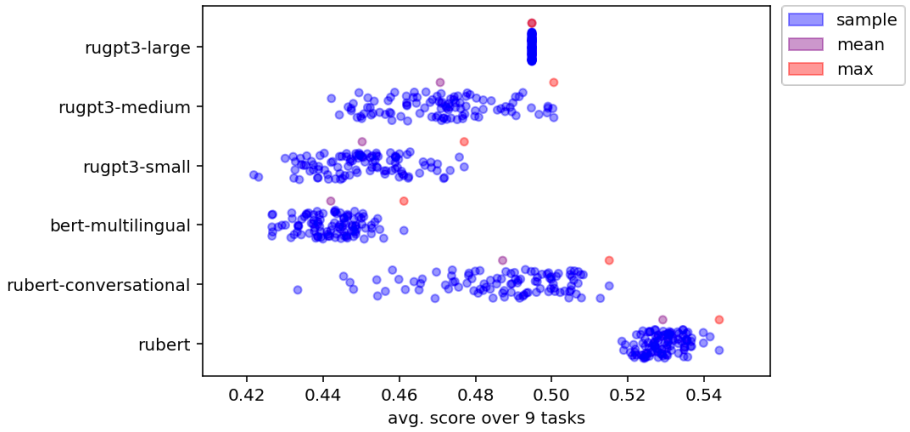

Fig. 2: Mean, maximum and averaged task-specific scores for the Russian SuperGLUE tasks.

5.2 Limitations

Our methodology inevitably has its limitations. First of all, it is fundamentally dependable on the length of the input sequences, once current Transformer language models have $O(n^2)$ in memory and computational time complexity. In the presented case, we use standardized datasets from SuperGLUE (for English) and RussianSuperGLUE. Although the datasets are fixed in the benchmarks they have different parameters. Tab. 2 shows average length in tokens for several datasets from RussianSuperGLUE. As one can see, the lowest value could be twice as small as the largest one. It is interesting to compare this statistics to Fig. 3. The seven lowest rows in it are single datasets listed in the mentioned table. We can see that the different models are perform instably on the datasets. We hypothesise that it is due to different lengths. Although once we compare the models on the combinations of the datasets (from two at once to all of them at once) the results became more and more stable.

Table 2: Statistics for selected datasets.

Dataset	RUSSE	TERRa	RCB	RWSD	MuSeRC	RuCoS	DaNetQA
Tokens per sample	12.11	18.46	13.63	14.93	19.76	20.55	21.02

Another important point is that we measure the highest memory consumption during the run. This decision allows us to compare the models in the worst conditions, thus providing the information important to run on a particular hardware piece with a limited memory. Although, the flaw of this method is that

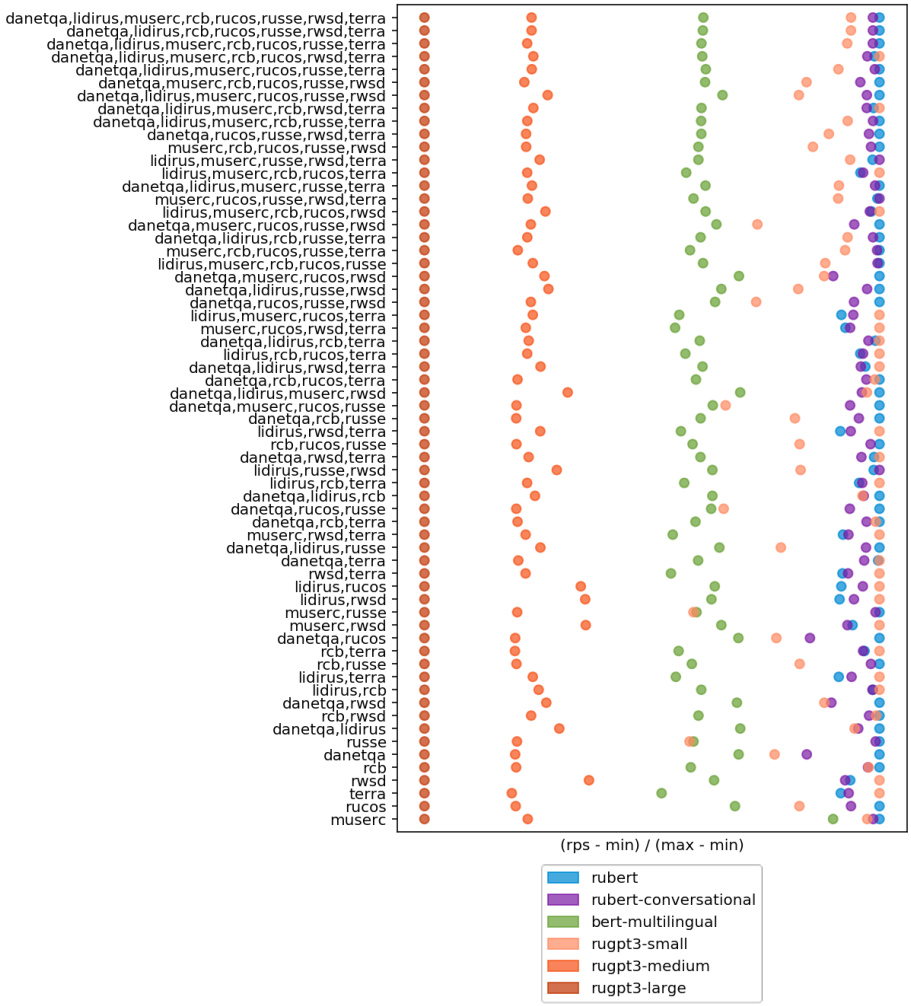

Fig. 3: Averaged inference speed for different combinations of the Russian SuperGLUE tasks.

the average memory consumption could be lower and the evaluated values will be overestimated.

5.3 Fitness Formulation

As stated in equation 1, there are three functions, namely f_Q, f_{Tp}, and f_M, which could affect the calculation. There are infinite possibilities for these functions, so we need to somehow limit our search with most standard and easy computable ones. We run a series of experiments with power function for all three, and

additionally logarithm for f_M. We fix two of three functions to be identity ones, while changing the third one.

Since Q is measured in range $[0, 1]$ it is natural somehow increase its contribution to final fitness value. But we put an exponent in range of $[-1, 1]$ to evaluate the fitness behaviour with different Q contributions. The resulting plots for RussianSuperGLUE models are presented at Fig. 4. As one can see on the figure, almost all the models are keeping their relative positions, except for RuGPT-3. This model takes second place in range $[-0.6, 0.5]$ and even the first one at $[-1, -0.6]$.

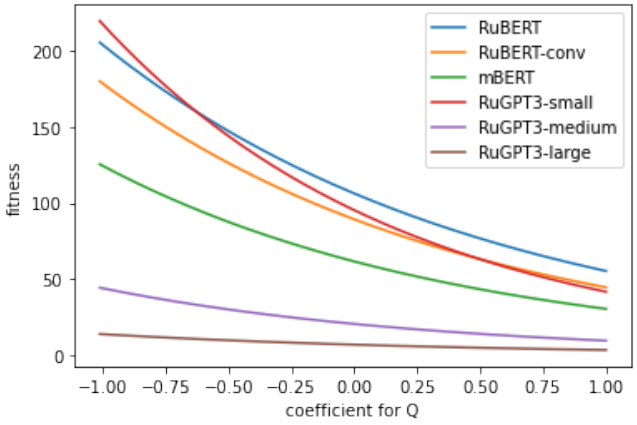

Fig. 4: f_Q function evaluation with different exponents.

Tp measures in natural numbers for the specific tasks, and cannot be less than 1. While aggregated it could be a real number, but still bigger than 1. For these numbers the ordering keeping the same for all the positive real exponents. The resulting plots for RussianSuperGLUE models are presented at Fig. 5. We keep only range $[0.75, 2.0]$ as the most informative.

As it stated in section 3.1, M is measured in bytes, which makes it effectively much larger than other two metrics. To handle this fact we apply normalization, dividing M by 2^{30}, thus M is represented in Gebibytes for our power function evaluation. The resulting plots for RussianSuperGLUE models are presented at Fig. 6. We also provide results for $f_M(\cdot) = \log(\cdot)$ as in equation 2. We scale the achieved results by factor of 15 to match the other presented values. As one can see in all the variants the existing ordering keeps the same.

Given these experiments we could conclude that the only important factor for evaluation is f_Q. We preferred put it to identity due to on the one hand computation efficiency and on the other hand the quality measuring conventionality, as discussed in Section 3.1.

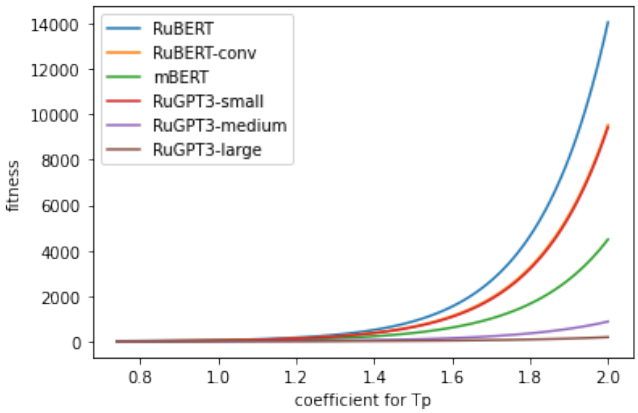

Fig. 5: $f_T p$ function evaluation with different exponents.

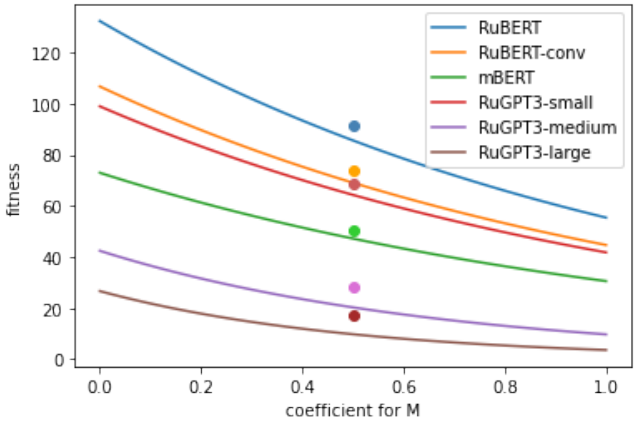

Fig. 6: f_M function evaluation with different exponents. The dots represent scaled values for logarithm function evaluation.

6 Conclusion

This work introduces the MOROCCO framework, which provides an assessment of language models' downstream quality combined with two computational efficiency metrics such as memory consumption and throughput during the inference stage. The proposed fitness metric allows to compose the GLUE-style leaderboards in a new way: to rank them so that the more high-precision, smaller and faster models are at the top, the accurate ones, but bigger and slower models are in the middle, and the most imprecise, largest and slowest ones are at the very bottom. Thus, to obtain a higher place on the leaderboard, researchers need to strive not for the score on the individual tasks, but also, develop optimal models in terms

of their practical use. A similar conditional assessment of the results has been mainly adopted for image classification and QA tasks. We expand this idea by integrating MOROCCO with the canonical SuperGLUE leaderboards showing the applicability for two languages. The presented framework is also compatible with the jiant framework and transformer models, making it easily applicable to evaluate a wide range of popular architectures, both multilingual and monolingual. We hope that our framework can be utilized in other jiant-based projects to provide a better and more detailed evaluation. This paper aims to stimulate the research on a compromise evaluation of the overall performance of NLP-models, which could be an alternative to the existing dominant "bigger is better" trend and would consider the problems of overfitting, over-parametrization, data redundancy, and many others.

A fruitful direction for future work is to cooperate with NLP developers and enthusiasts to further search for the most optimal solutions, including organizing the competition of multilingual NLP models on existing benchmarks as a possible step. Another line of work includes extending the framework with other metrics such as time and memory use required for fine-tuning, the time needed to achieve the best quality, and robustness towards task-specific adversarial attacks.

Bibliography

Fuad Aleskerov, Denis Bouyssou, and Bernard Monjardet. 2007. Utility maximization, choice and preference, volume 16. Springer Science & Business Media.

Alexandra Birch, Andrew Finch, Minh-Thang Luong, Graham Neubig, and Yusuke Oda. 2018. Findings of the second workshop on neural machine translation and generation. In Proceedings of the 2nd Workshop on Neural Machine Translation and Generation, pages 1–10.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. pages 4171–4186.

Kawin Ethayarajh and Dan Jurafsky. 2020. Utility is in the eye of the user: A critique of nlp leaderboards. arXiv preprint arXiv:2009.13888.

Hiroaki Hayashi, Yusuke Oda, Alexandra Birch, Ioannis Konstas, Andrew Finch, Minh-Thang Luong, Graham Neubig, and Katsuhito Sudoh. 2019. Findings of the third workshop on neural generation and translation. In Proceedings of the 3rd Workshop on Neural Generation and Translation, pages 1–14.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention.

Kenneth Heafield, Hiroaki Hayashi, Yusuke Oda, Ioannis Konstas, Andrew Finch, Graham Neubig, Xian Li, and Alexandra Birch. 2020. Findings of the fourth workshop on neural generation and translation. In Proceedings of the Fourth Workshop on Neural Generation and Translation, pages 1–9.

Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. 2020. Towards the systematic reporting of the energy and carbon footprints of machine learning. Journal of Machine Learning Research, 21(248):1–43.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In International Conference on Machine Learning, pages 4411–4421. PMLR.

Yuri Kuratov and Mikhail Arkhipov. 2019. Adaptation of deep bidirectional multilingual transformers for russian language. arXiv preprint arXiv:1905.07213.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations.

Yaobo Liang, Nan Duan, Yeyun Gong, Ning Wu, Fenfei Guo, Weizhen Qi, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, Xiaodong Fan, Ruofei Zhang, Rahul Agrawal, Edward Cui, Sining Wei, Taroon Bharti, Ying Qiao, Jiun-Hung Chen, Winnie Wu, Shuguang Liu, Fan Yang, Daniel Campos, Rangan Majumder, and Ming Zhou. 2020. http://arxiv.org/abs/2004.01401 Xglue: A new benchmark dataset for cross-lingual pre-training, understanding and generation.

Qi Liu, Matt J. Kusner, and Phil Blunsom. 2020. http://arxiv.org/abs/2003.07278 A survey on contextual embeddings.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

Zhiyi Ma, Kawin Ethayarajh, Tristan Thrush, Somya Jain, Ledell Wu, Robin Jia, Christopher Potts, Adina Williams, and Douwe Kiela. 2021. Dynaboard: An evaluation-as-a-service platform for holistic next-generation benchmarking. arXiv preprint arXiv:2106.06052.

Sewon Min, Jordan Boyd-Graber, Chris Alberti, Danqi Chen, Eunsol Choi, Michael Collins, Kelvin Guu, Hannaneh Hajishirzi, Kenton Lee, Jennimaria Palomaki, Colin Raffel, Adam Roberts, Tom Kwiatkowski, Patrick Lewis, Yuxiang Wu, Heinrich Küttler, Linqing Liu, Pasquale Minervini, Pontus Stenetorp, Sebastian Riedel, Sohee Yang, Minjoon Seo, Gautier Izacard, Fabio Petroni, Lucas Hosseini, Nicola De Cao, Edouard Grave, Ikuya Yamada, Sonse Shimaoka, Masatoshi Suzuki, Shumpei Miyawaki, Shun Sato, Ryo Takahashi, Jun Suzuki, Martin Fajcik, Martin Docekal, Karel Ondrej, Pavel Smrz, Hao Cheng, Yelong Shen, Xiaodong Liu, Pengcheng He, Weizhu Chen, Jianfeng Gao, Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Wen tau Yih. 2021. http://arxiv.org/abs/2101.00133 Neurips 2020 efficientqa competition: Systems, analyses and lessons learned.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How Multilingual is Multilingual BERT? pages 4996–5001.

Yada Pruksachatkun, Phil Yeres, Haokun Liu, Jason Phang, Phu Mon Htut, Alex Wang, Ian Tenney, and Samuel Bowman. 2020. jiant: A software toolkit for research on general-purpose text understanding models. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 109–117.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. OpenAI blog, 1(8):9.

Anna Rogers. 2019. https://hackingsemantics.xyz/2019/leaderboards/ How the transformers broke nlp leaderboards.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In The 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing.

Tatiana Shavrina, Alena Fenogenova, Emelyanov Anton, Denis Shevelev, Ekaterina Artemova, Valentin Malykh, Vladislav Mikhailov, Maria Tikhonova, Andrey Chertok, and Andrey Evlampiev. 2020. Russiansuperglue: A russian language understanding evaluation benchmark. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 4717–4726.

Tatiana Shavrina and Valentin Malykh. 2021. How not to lie with a benchmark: Rearranging nlp learderboards. In ICBINB@NeurIPS 2021.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In China National Conference on Chinese Computational Linguistics, pages 194–206. Springer.

Henry Tsai, Jason Riesa, Melvin Johnson, Naveen Arivazhagan, Xin Li, and Amelia Archer. 2019. Small and practical bert models for sequence labeling. In EMNLP/IJCNLP (1).

C. J. van Rijsbergen. 1979. Information Retrieval. Butterworth-Heinemann.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. pages 5998–6008.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. Advances in Neural Information Processing Systems, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. https://doi.org/10.18653/v1/W18-5446 GLUE: A multi-task benchmark and analysis platform for natural language understanding. pages 353–355.

Alex Wang and Thomas Wolf. 2020. Overview of the sustainlp 2020 shared task. In Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing, pages 174–178.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2020. http://arxiv.org/abs/1905.12616 Defending against neural fake news.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tieyan Liu. 2019. Incorporating bert into neural machine translation. In International Conference on Learning Representations.

A Task Description

The experiments are run on a diverse set of 9 tasks[10] from the SuperGLUE benchmarks for each language (see Table 3): Recognizing Textual Entailment (RTE) task is aimed to capture textual entailment in a binary classification form; Commitment Bank belongs to the natural language inference (NLI) group of tasks type with a 3-way classification; Diagnostic dataset which is another test set for the RTE task annotated with various linguistic and semantic phenomena; Words in Context task is based on word sense disambiguation problem in a binary

[10] SuperGLUE benchmark also includes an additional Winogender Schema Diagnostics task which is a dataset which we do not consider in the experiments since it is not included in Russian SuperGLUE.

classification form; Choice of Plausible Alternatives is a binary classification task aimed at accessing commonsense causal reasoning; Yes/No Questions is a binary QA task for closed questions; Multi-Sentence Reading Comprehension is a task on multi-hop machine reading comprehension (MRC); Reading Comprehension with Commonsense Reasoning is an MRC task, where it is required to fill the masked gaps in the sentence with the best fitting entities from the given text paragraph; Winograd Schema Challenge is devoted to co-reference resolution in a binary classification form.

Table 3: Datasets statistics. MCC stands for Matthews' Correlation Coefficient; Acc - Accuracy; EM - Exact Match. The size train/validation/test splits are provided in "Samples" columns.

Task Type	Task	SuperGLUE		Russian SuperGLUE		Metric
		Name	Samples	Name	Samples	
NLI	Recognizing Textual Entailment	RTE	2490/277/3000	TERRa	2616/307/3198	Acc
	Commitment Bank	CB	250/56/250	RCB	438/220/438	Avg. F1 / Acc
NLI & diagnostics	Diagnostic	AX-b	0/0/1104	LiDiRus	0/0/1104	MCC
Common Sense	Words in Context	WiC	5428/638/1400	RUSSE	19845/8508/18892	Acc
	Choice of Plausible Alternatives	COPA	400/100/500	PARus	400/100/500	Acc
World Knowledge	Yes/No Questions	BoolQ	9427/3270/3245	DaNetQA	1749/821/805	Acc
Machine Reading	Multi-Sentence Reading Comprehension	MultiRC	456/83/166	MuSeRC	500/100/322	F1 / EM
	Reading Comprehension with Commonsense Reasoning	ReCoRD	65709/7481/7484	RuCoS	72193/7577/7257	F1 / EM
Reasoning	The Winograd Schema Challenge	WSC	554/104/146	RWSD	606/204/154	Acc

B Standalone Run

The user needs to clone the project repository to their machine to run the framework locally. MOROCCO works with the Docker container engine and provides the corresponding code. We consider the following procedure for the evaluation: fine-tune a model for a specific task, build a Docker container with the model, run the container on the test data to get the outputs, collect the outputs for multiple runs and conduct the evaluation. The downstream performance can be received by submitting to the corresponding leaderboard.

For instance, the fine-tuning (training) the RuBERT model for RUSSE could be done with this command:

```
python main.py train rubert russe \
~/path/for/logs ~/data/RUSSE
--seed=3
```

Note that this run uses the fixed random seed which can be adjusted.

To infer the trained model for the specific task, run the following code snippet:

```
python main.py infer \
~/path/for/logs/rubert/ russe \
--batch-size=32
```

To build the Docker container with the trained model, run the following code snippet:

```
python main.py docker build \
~/path/for/logs/rubert/ russe \
rubert-russe
```

To infer the container with the model, storing its outputs, run the following code snippet:

```
docker run --gpus all \
--interactive --rm rubert-russe \
--batch-size 8 \
<~/data/RUSSE/val.jsonl \
>preds.jsonl
```

To evaluate the model by the task-specific metrics, submit your model predictions to the leaderboard or run the following code snippet on the validation set (preliminarily making the predictions):

```
python main.py eval russe \
preds.jsonl \
~/data/RUSSE/val.jsonl
```

Finally, to get the memory footprint and inference speed results, run the following code snippet:

```
for index in 01 02 03 04 05;
  do python main.py docker \
  bench rubert-russe ~/data \
  russe --input-size=2000 \
  --batch-size=32 \
  >~/benches/rubert/\
  russe/2000_32_$index.jl;
done
```

Submission to RussianSuperGLUE To make a submission for the leaderboard, you need to push a Docker container to a Docker repository, login to RussianSuperGLUE website, and add a link to the Docker container you created into the submission form. Figure ?? presents an example of the leaderboard submission (multilingual BERT) reported by the introduced computational efficiency metrics.

C Исправления по замечаниям

Reviewer 1.

> First of all, there is a violation of format: the maximum number of pages is 15, while this paper has 17. Next, the paper is not in an LNCS format.

Dear R1, we have ensured the format and number of pages.

> Next, the quality of writing is poor and makes the paper hard to understand...

Thank you, we have improved the description in section 3.1.

> 1) The motivation of such ranking is unclear...

> 2) Model looks shallow and the choice of underlying metrics is not justified...

We have improved section 3.1 with more clear motivation. Also we have added section 5.2 where we discuss the limitations we see for our methodology.

We have mostly incorporated the minor remarks.

Reviewer 2.

> It could be interesting to include in the comparison in the paper some more novel models, for example RuRoBERTa.

Thank you, we have checked a few models to not overwhelm the paper (and hence a reader) with plethora of numbers. But the leaderboard is open and we hope that the community will be filling it.

Reviewer 3.

> Методология измерения потребления памяти представляется недостаточно проработанной...

Мы согласны с замечанием, что методология потребления памяти может быть улучшена. Но наша методология была выбрана за ее простоту в измерениях. Потенциальные проблемы с неоднозначностью результатов, как мы надеимся, решаются с помощью статистики, а именно усреднения их по нескольким запускам (5 в наших экспериментах). Относительно той части замечания, что есть различные по длине тексты, это справедливо, однако отчасти учтено в нашей методологии. Мы изначально рассматриваем композицию из результатов запусков на разных по своим характеристикам задачам. Это освещено в секции 5.1, плюс мы добавили секцию 5.2, где постарались описать известные нам ограничения нашей методологии.

> При тестировании скорости вывода, из текста статьи непонятно, повторялись прогоны на одном и том же наборе данных или на разных?

Спасибо за замечание, да запросы прогоняются на одних и тех же данных, чтобы усреднять влияние внеших по отношению к данным факторам (например, загрузки подсистемы ввода-вывода).

P.S. При включении русского языка, к сожалению, ломается форматирование, в частности, пропадает выделение жирным шрифтом. Это будет исправлено в финальной версии работы.