MOROCCO: Model Resource Comparison Framework

Anonymous ACL submission

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 performance as a primary metric which limits the scope of model comparison in terms of their practical use. This paper presents MOdel ResOurCe COmparison (MOROCCO), a publicly available framework\(^1\) that allows to assess models with respect to their downstream quality combined with two computational efficiency metrics such as memory consumption and throughput during the inference stage. The framework allows for a flexible integration with popular leaderboards compatible with jiant environment that supports over 50 downstream tasks. We demonstrate the MOROCCO applicability by evaluating 10 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. A large number of 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 number of parameters and the architecture design (Liu et al., 2020) has been mainly assessed by means of downstream performance as a primary metric on many common benchmarks such as GLUE (Wang et al., 2018), XGLUE (Liang et al., 2020), SuperGLUE (Wang et al., 2019) and XTREME (Hu et al., 2020). Despite the fact that the benchmarks provide a standard for a direct model comparison, the performance-oriented approach limits the scope of the evaluation methods (Ethayarajh and Jurafsky, 2020). Understanding the need of expanding the methodology, various benchmarks and contests have been proposed targeting computational and technical aspects of the models (see Section 2), with the problem of continuously growing number of parameters highlighted (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 standalone framework that aims at measuring both the downstream performance and computational efficiency of the models in a fixed environment. Second, MOROCCO can be potentially integrated with popular leaderboards compatible with jiant environment (Pruksachatkun et al., 2020) that supports over 50 downstream tasks\(^2\), including GLUE-style ones. We demonstrate the MOROCCO applicability by evaluating 10 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.

\(^{1}\)The url will be provided upon acceptance.

\(^{2}\)https://github.com/nyu-mll/jiant/blob/master/guides/tasks/supported_tasks.md
specifically those that meet the user 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). The benchmark infrastructure involves a set of downstream tasks and a public leaderboard. A submission to the leaderboard consists of predictions made by the user model on the publicly available test sets, and is further evaluated by the task-specific metrics. The more recent benchmarks follow the same evaluation procedure but aim at domain-specific areas, such as dialogue systems (Mehri et al., 2020), biomedical NLU and reasoning (Gu et al., 2020), or at evaluation in the cross-lingual setting (Liang et al., 2020; Hu et al., 2020). Such evaluation method does not consider any computational and technical aspects of the models that differ greatly by 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 which limits the reproducibility of the leaderboard results (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) challenged the participants to create an effective NLP-system for open-domain question answering (ODQA). The submissions are limited by a number of performance and technical requirements which stimulate the community to develop optimal ODQA systems that can achieve prominent performance while satisfying the technical needs and operating on an optimal amount of retrieval corpora.

The SustaiNLP challenge (Wang and Wolf, 2020) was aimed at developing efficient but yet accurate models. The efficiency is estimated as the power consumed throughout the inference time calculated by means of 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 on average around 2 absolute points lower.

DynaBench (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 datasets.

Last but not least, DAWNBench (Coleman et al., 2017) measures the end-to-end image classification and QA systems reporting time required to achieve a particular performance score, as well as the downstream performance itself.

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 HuggingFace library (Wolf et al., 2019).

Submission details  To conduct the evaluation of the model’s performance on the RussianSuperGLUE tasks, a team 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. To eliminate the running time and memory footprint dispersion caused by technical reasons, we perform several runs and compute the median values. Next, the output from the container is evaluated with the task-specific metrics. The results are used to compute the final evaluation score for the whole submission. To ensure the compa-
rability of the collected metrics, we fix the com-
putation hardware. We use Yandex.Cloud virtual
instances, where the following hardware is guaran-
teed: 1 × Intel Broadwell CPU, 1 × NVIDIA Tesla
V100 GPU. The Docker containers are equipped
with Ubuntu 20.04. Following the SuperGLUE in-
frastructure, our framework is designed to comprise
with jiant framework, alongside with simple re-
quirements for the evaluation containers built upon
other frameworks, and can be run locally using the
code base.

3.1 Metrics
We report the computational efficiency of the tested
model by means of the memory footprint and infer-
ence speed.

Memory footprint allows to account for the
model’s size and the number of weights implicitly,
as there is strong dependency. To measure model
GPU RAM usage $M$ we run a container with a
single record as input, measure the maximum GPU
RAM consumption, repeat the procedure 5 times
and compute a median value.

Inference speed measures directly how much time
the model consumes on a specific hardware, es-
suming implicitly the model’s complexity. To
measure the inference speed $T_p$ we run a container
with $N$ records as input, with batch size 32.
We also estimate initialization time $T_{init}$ with running
a container with an input of size 1. Inference speed
$T_p$ is computed as follows: $T_p = \frac{N}{T_N - T_{init}}$. In our
experiments we use $N = 2000$ which can be ad-
justed by the user. We repeat the procedure 5 times
to compute a median value.

Overall, our evaluation procedure utilizes three
different scores, namely the task-specific perfor-
ance score $Q$, the inference speed $T_p$ and the
memory footprint $M$. We propose to take into ac-
count these three characteristics of a model and make an integral measure of its “fitness” $F$ that combines task-specific and computational metrics:

$$F = Q \times \frac{T_p}{\log(M)}$$

where $Q$ is the metric-based score for a specific
task, $M$ is measured in bytes, $T_p$ is measured in
records per second (RPS). We take a logarithm of

$M$ since the model size increase is exponential for
the modern models (Sanh et al., 2019). This mea-
sure is motivated by the common idea that memory
consumption should be lowered, while the achieved
quality and processing speed should be increased
(Henderson et al., 2020).

3.2 Tasks
The experiments are run on a diverse set of 9 tasks\footnote{SuperGLUE benchmark also includes additional Wino-
gerder Schema Diagnostics task which is a dataset which we
do not consider in the experiments since it is not included in
Russian SuperGLUE.} from the SuperGLUE benchmarks for each lan-
guage (see Table 1): Recognizing Textual Entail-
ment (RTE) task is aimed to capture textual entail-
ment in a binary classification form; Commitment
Bank belongs to the natural language inference
(NLI) group of tasks type with a 3-way classifica-
tion; Diagnostic dataset which is in fact another
test set for the RTE task annotated with various lin-
guistic and semantic phenomena; Words in Con-

\footnote{https://cloud.yandex.com/}

\footnote{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 needs.}
### Table 1: Datasets statistics. MCC stands for Matthews’ Correlation Coefficient; Acc - Accuracy; EM - Exact Match. The size train/validation/test splits are provided in “Samples” columns.

<table>
<thead>
<tr>
<th>Task Type</th>
<th>Task</th>
<th>SuperGLUE</th>
<th>Russian SuperGLUE</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Name</td>
<td>Samples</td>
<td>Name</td>
</tr>
<tr>
<td>NLI</td>
<td>Recognizing Textual Entailment Commitment Bank</td>
<td>RTE</td>
<td>2490/277/3000</td>
<td>TERRa</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CB</td>
<td>250/56/250</td>
<td>RCB</td>
</tr>
<tr>
<td>NLI &amp; diagnostics</td>
<td>Diagnostic</td>
<td>AX-b</td>
<td>0/0/1104</td>
<td>LiDiRus</td>
</tr>
<tr>
<td>Common Sense</td>
<td>Words in Context</td>
<td>WiC</td>
<td>5428/638/1400</td>
<td>RUSSE</td>
</tr>
<tr>
<td></td>
<td>Choice of Plausible Alternatives</td>
<td>COPA</td>
<td>400/100/500</td>
<td>PARus</td>
</tr>
<tr>
<td>World Knowledge</td>
<td>Yes/No Questions</td>
<td>BoolQ</td>
<td>9427/3270/3245</td>
<td>DaNetQA</td>
</tr>
<tr>
<td>Machine Reading</td>
<td>Multi-Sentence Reading Comprehension</td>
<td>MultiRC</td>
<td>456/83/166</td>
<td>MuSeRC</td>
</tr>
<tr>
<td></td>
<td>Multi-Sentence Reading Comprehension with Commonsense Reasoning</td>
<td>ReCoRD</td>
<td>65709/7481/7484</td>
<td>RuCoS</td>
</tr>
<tr>
<td>Reasoning</td>
<td>The Winograd Schema Challenge</td>
<td>WSC</td>
<td>554/104/146</td>
<td>RWSD</td>
</tr>
</tbody>
</table>

The text task is based on word sense disambiguation problem in a binary 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.

### 3.3 Models

We run the experiments on the following publicly available models that achieved competitive performance on both 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-3⁶ (ru-gpt3-small, ru-gpt3-medium, and ru-gpt3-large), RuBERT-base (rubert) (Kuratov and Arkhipov, 2019), and Conversational RuBERT-base⁷ (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 $Q$, $T_p$, and $M$ (see Section 3.1). These figures discover Pareto frontiers for both languages. For English, GPT-2, mono- and multilingual BERT models and RoBERTa appear to be Pareto-optimal. For Russian, ruGPT3-large, ruGPT3-medium, ruBERT and Conversational ruBERT dominate other models according to the Pareto rule.

The fitness metric $F$ results are presented in Table 2. RoBERTa model had shown the best score for English, while RuBERT is the best fit among the tested models for Russian. 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

---

⁶https://github.com/sberbank-ai/ru-gpts
⁷https://huggingface.co/DeepPavlov/rubert-base-cased-conversational
askew towards the English language. Overall, the
evaluation results have revealed better models by
means of task-specific quality, memory footprint,
and inference speed.

<table>
<thead>
<tr>
<th>English</th>
<th>Russian</th>
</tr>
</thead>
<tbody>
<tr>
<td>en_bert_base</td>
<td>rubert</td>
</tr>
<tr>
<td>5.05</td>
<td>4.84</td>
</tr>
<tr>
<td>bert-multilingual</td>
<td>bert-multilingual</td>
</tr>
<tr>
<td>4.79</td>
<td>3.30</td>
</tr>
<tr>
<td>en_roberta_base</td>
<td>rubert-conversational</td>
</tr>
<tr>
<td>6.63</td>
<td>4.59</td>
</tr>
<tr>
<td>albert</td>
<td>rugpt3-small</td>
</tr>
<tr>
<td>5.41</td>
<td>3.89</td>
</tr>
<tr>
<td>en_gpt2</td>
<td>rugpt3-medium</td>
</tr>
<tr>
<td>1.95</td>
<td>1.89</td>
</tr>
<tr>
<td></td>
<td>rugpt3-large</td>
</tr>
<tr>
<td></td>
<td>1.24</td>
</tr>
</tbody>
</table>

Table 2: Fitness evaluation for the models in English
and Russian.

4.1 Discussion

Averaging the estimates of $Q$, $Tp$, and $M$ is one of
the main limitations of proposed evaluation procedure.
Averaging memory consumption $M$ is less
problematic, as it is relatively stable for any rea-
sonable sample size. However two other metrics
require more detailed investigation. Figure 2 com-
pare the mean and maximum values of $Q$ with re-
spect to different models. Each model was trained
five times with different random seeds and was
scored ten times, which makes overall fifty runs.
The only exception was made to the largest model,
rugpt3-large, which was trained only ones. 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 for all runs. The
ranking, achieved by maximum and mean scores is
same.

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

Figure 3 compares averaged normalized infer-
ence 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 rank-
ing remains mostly unchanged, while occasionally
top models exchange positions.

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

5 Standalone Run

To run our framework locally you need to clone the
project repository first to your own machine. MO-
ROCCO works with the Docker container engine
and provides the corresponding code. We consider
the following procedure for the evaluation: train 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 per-
formance can be received by making a submission
on the corresponding leaderboard.

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

```python
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 \
>~/benches/rubert/ \
>~/path/for/logs/rubert/ russe \
>preds.jsonl

To evaluate the model by the task-specific metrics, make a submission with your model predictions to the leaderboard or run the following code snippet on the validation set (preliminarily making predictions for the set):
python main.py eval russe \
preds.jsonl \
~/data/RUSSE/val.jsonl

Finally, to get the results for the memory footprint and inference speed, 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

6 Conclusion

This work introduces the MOROCCO framework which provides assessment of language models with respect to their downstream quality combined with two computational efficiency metrics such as memory consumption and through-put 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 most high-precision, smallest and fastest models are in 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 at stimulating 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 take into account the problems of overfitting, over-parametrization, data redundancy, and many others.

A fruitful direction for future work is cooperation 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, time needed to achieve the best quality, and robustness towards task-specific adversarial attacks.

References


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

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, Aman- 
preet Singh, Julian Michael, Felix Hill, Omer Levy, 
and Samuel R Bowman. 2019. Superglue: A stick- 
ier benchmark for general-purpose language under- 
standing systems. Advances in Neural Information 
Processing Systems, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix 
Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: 
A multi-task benchmark and analysis platform for 

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

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

Rowan Zellers, Ari Holtzman, Hannah Rashkin, 
Yonatan Bisk, Ali Farhadi, Franziska Roesner, and 
Yejin Choi. 2020. Defending against neural fake 
news.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, 
Incorporating bert into neural machine translation. In 
International Conference on Learning Representa- 
tions.