EffEval: A Comprehensive Evaluation of Efficiency for MT Evaluation Metrics

Daniil Larionov[1], **Jens Grünwald**[2], **Christoph Leiter**[1], **Steffen Eger**[1]
[1] Department of Computer Science, [1] Natural Language Learning Group, Faculty of Technology
[2] Technical University of Darmstadt, [1] Bielefeld University
daniil.larionov@uni-bielefeld.de

Abstract

Efficiency is a key property to foster inclusiveness and reduce environmental costs, especially in an era of LLMs. In this work, we provide a comprehensive evaluation of efficiency for MT evaluation metrics. Our approach involves replacing computation-intensive transformers with lighter alternatives and employing linear and quadratic approximations for alignment algorithms on top of LLM representations. We evaluate six (reference-free and reference-based) metrics across three MT datasets and examine 16 lightweight transformers. In addition, we look into the training efficiency of metrics like COMET by utilizing adapters. Our results indicate that (a) TinyBERT provides the optimal balance between quality and efficiency, (b) CPU speed-ups are more substantial than those on GPU; (c) WMD approximations yield no efficiency gains while reducing quality and (d) adapters enhance training efficiency (regarding backward pass speed and memory requirements) as well as, in some cases, metric quality. These findings can help to strike a balance between evaluation speed and quality, which is essential for effective NLG systems. Furthermore, our research contributes to the ongoing efforts to optimize NLG evaluation metrics with minimal impact on performance. To our knowledge, ours is the most comprehensive analysis of different aspects of efficiency for MT metrics conducted so far.

1 Introduction

Evaluation is crucial to progress in fields such as NLP and machine learning, as it is used to identify and assess the most promising, state-of-the-art approaches. It is particularly challenging for Natural Language Generation (NLG) systems as text generation is open-ended: multiple outputs, with very different surface-level realizations, can be equally correct (Celikyilmaz et al., 2020). This insight makes classical lexical overlap metrics such as BLEU (Papineni et al., 2002) or ROUGE (Lin, 2004) unsuitable as high-quality evaluation metrics. Consequently, there has been a recent surge of interest (Freitag et al., 2022) in developing evaluation metrics based on pretrained large language models (LLMs), which can better cope with lexical variation, thus yielding metrics that correlate much better with human assessments of quality. Notable examples are MoverScore (Zhao et al., 2019), BERTScore (Zhang et al., 2020), and BARTScore (Yuan et al., 2021).

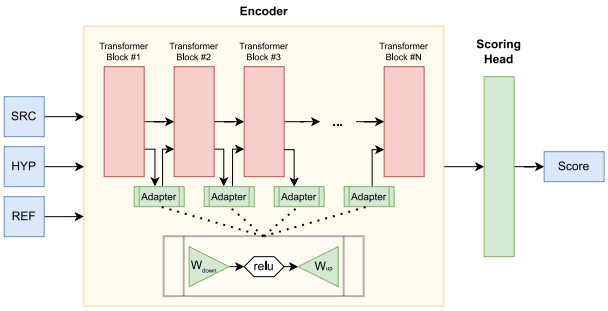

Figure 1: The COMET metric with *pfeiffer* adapter configuration. Parameters of *red* blocks remain frozen during training, while parameters of *green* blocks are optimized.

However, basing evaluation metrics on large transformers (and thus boosting the quality of the metrics) also has downsides: for example, the associated computational burden (i) may promote inequality among researchers, hindering diversity, as not everyone has access to expensive compute resources[1] and (ii) incurs high environmental costs, one of the most critical issues of our time (Strubell et al., 2019). (iii) Inefficient metrics – which are in addition non-transparent (Leiter et al., 2022) – may also prevent high-quality metrics from being deployed by the community, a potential reason why older, lower-quality but faster metrics such as

[1]E.g., Kamal Eddine et al. (2022) mention that they cannot run variants of BERTScore on a 12GB GPU and the situation would even be worse for more disadvantaged scholars around the world.

Findings of the Association for Computational Linguistics: EMNLP 2023, pages 78–96
December 6-10, 2023

BLEU are still popular (Marie et al., 2021).

To better illustrate the issue, let us consider a typical setup for evaluating machine translation (MT) systems (e.g., the setup of WMT shared tasks). In such a case, one may, for example, have 30k segments to evaluate per language pair, 5 different language pairs, and 50 assessed MT systems. If one uses the BERTScore (Zhang et al., 2020) metric with the author-suggested RoBERTa-Large (Liu et al., 2020) encoder, then it would take 71 hours to completely evaluate all the MT systems on a single Nvidia A100 GPU (given that users do have access to GPUs), producing around 8kg CO_2-eq of carbon footprint. If access to the GPU is restricted, one could run the metric on a CPU, but it would take more than 950 hours to do a full evaluation, producing 5.4kg CO_2-eq of carbon footprint. While in the case of a shared task/challenge, the evaluation of model outputs is usually not frequent, there are some cases where evaluation is done constantly, such as hyperparameter search and neural architecture search.

Apart from evaluating MT systems, evaluation metrics have a variety of possible use cases that would greatly benefit from the computationally efficient solutions: **a)** metrics can be used as reward functions in Reinforcement Learning pipelines; **b)** some metrics can be used in the filtering of massive, web-crawled parallel corpora; **c)** they can be used in an online setting for real-time re-ranking of MT systems outputs.

Thus, developing *light-weight high-quality* evaluation metrics for NLG is imperative, which we explore in depth in this work. We focus on MT as a prime instance of NLG, which also yields a diverse set of scenarios for efficiency, including training efficiency and the efficiency of multilingual models. Nonetheless, we believe that our insights hold more generally.

Our paper about **Eff**icient **Eval**uation (EffEval) presents the following main contributions:

• We provide a comprehensive analysis of inducing efficient, high-quality evaluation metrics based on three principles: (i) replacing a computation-heavy transformer in the metrics by much smaller ones, obtained e.g. via pruning or distillation; (ii) replacing costly alignment techniques (Word Mover Distance; WMD) on top of transformers with cheaper approximations; (iii) implementing parameter-efficient training with adapters.

• Our analysis comprises three MT datasets, six evaluation metrics, and 16 light-weight transformers as replacements for the original transformers.

• Based on our large-scale analysis, we find that: (a) for each metric, there is often at least one efficient transformer which leads to higher quality and higher efficiency at the same time, but on average, there is a drop in quality when employing more efficient models; (b) for example, for "semantic similarity" metrics like BERTScore, we find that the distilled transformer TinyBERT (Jiao et al., 2020) has the best performance-quality trade-off — on average, it retains 97% of the original quality while being 5x faster at inference time; (c) speedups differ substantially on CPU vs. GPU; (d) WMD approximations yield no efficiency gains in our experiments (as WMD itself is less costly than embedding computation), but have adverse effects on quality in 2 out of 3 datasets.

• Furthermore, we investigate training efficiency – a crucial aspect of recent MT metrics which leverage more and more supervision signals (Rei et al., 2020), despite criticisms (Belouadi and Eger, 2023) – by examining the performance of the popular COMET (Rei et al., 2020, 2022a) and COMET-INHO (Rei et al., 2022b) trainable metrics when utilizing adapters (Houlsby et al., 2019). Our findings indicate that adapters contribute to an increased backward pass speed by 37%-102% and a 26%-32% reduction in memory usage, depending on the model variant. Along with gains on training performance, adapter-enabled models have outperformed the fully-trainable ones, while being trained on the same amount of data.

Our code is available at `https://github.com/NL2G/effeval`.

2 Related work

Our work connects to (1) transformer-based evaluation metrics and to (2) efficiency. Here, we provide only a brief overview of the related papers. Appendix A contains additional related work.

Evaluation metrics: Recent transformer-based metrics utilize BERT-based models like BERTScore (Zhang et al., 2020) and MoverScore (Zhao et al., 2019). Extensions include BARTScore (Yuan et al., 2021), which reads off probability estimates as metric scores directly from text generation systems, and MENLI (Chen and Eger, 2023), which uses probabilities from models fine-tuned on Natural Language Inference task. These metrics are reference-based (comparing the

MT output to a human reference), like BERTScore and MoverScore, or reference-free (comparing the MT output to the source text), like XMoverScore (Zhao et al., 2020) and SentSim (Song et al., 2021), and some are trained (fine-tuned on human scores) like COMET (Rei et al., 2020) while others are untrained, like BERTScore. Trained metrics typically show higher correlations with human assessments, but require more resources and, thus are more costly. Transversal approaches by Fu et al. (2023); Liu et al. (2023) use LLM predictions.

Efficiency: Techniques like knowledge distillation (Hinton et al., 2015; Ganesh et al., 2021), dynamic inference acceleration (Sun et al., 2019; Xin et al., 2020; Zhu, 2021), and adapters (Pfeiffer et al., 2020a; Houlsby et al., 2019) seek to improve model efficiency. Knowledge distillation involves a smaller student model learning from a larger teacher, e.g., DistilBERT (Sanh et al., 2019) and TinyBERT (Jiao et al., 2020). Kamal Eddine et al. (2022) distill an efficient evaluation metric called FrugalScore from the teachers BERTScore/MoverScore. Dynamic inference acceleration adds early exit ramps based on representation changes in encoder layers. Adapters freeze pre-trained transformers and train intermediate layers, which can reduce memory usage and improve training speed with varying complexity (Pfeiffer et al., 2020b; He et al., 2022; Liu et al., 2022).

3 Approach

For optimizing the metrics, we explore three approaches: (i) We replace transformers with smaller and more efficient variants in §3.1 and (ii) we accelerate token matching by calculating Word Centroid Distance (WCD) and Relaxed Word Mover's Distance (RWMD) instead of more computationally complex WMD for MoverScore and XMoverScore in §3.2. Finally, (iii) we explore the impact of using adapters on training efficiency and metric quality for COMET and COMETINHO.

3.1 Replacing transformer models

Similarity-based metrics like BERTScore (Zhang et al., 2020), (X)MoverScore (Zhao et al., 2019, 2020), BaryScore (Colombo et al., 2021) and SentSim (Song et al., 2021) are not dependent on specific models for calculating token representations. These metrics can leverage any model that generates contextualized vector representations for the input text tokens. By default, the authors of BERTScore suggest using RoBERTa-Large (Liu et al., 2020), which is computationally expensive. To investigate the impact of utilizing more efficient transformer models, we replace default encoders with pruned, distilled, and dynamically accelerated ones:

Distillation: We use DistilBERT (Sanh et al., 2019) and TinyBERT (Jiao et al., 2020) for reference-based semantic similarity metrics BERTScore, MoverScore and BaryScore; dBART (Shleifer and Rush, 2020) for reference-based BARTScore; multilingual DistilMBERT (Sanh et al., 2019) and XtremeDistil (Mukherjee and Hassan Awadallah, 2020) for reference-free semantic similarity metrics XMoverScore and SentSim; DistilGPT-2 (von Platen, 2022) for XMoverScore and mMiniLM (Wang et al., 2021) for XMoverScore and SentSim. **Pruning**: We also examine the performance of one of the miniature BERT models, $\text{BERT}_{\text{TINY}}$, introduced in (Turc et al., 2020), with BERTScore, MoverScore and BaryScore. **Dynamic Inference Acceleration**: For BERTScore, MoverScore, and BaryScore, we build a version using DeeBERT's early exiting (Xin et al., 2020).

3.2 Improving Token Matching Efficiency

Our second approach for building more efficient metrics involves enhancing the token matching speed in metrics, specifically focusing on WMD in MoverScore (Zhao et al., 2019, 2020; Colombo et al., 2021), a popular approach for token matching in evaluation metrics. Proposed by Kusner et al. (2015), WMD is a specialized version of EMD (Rubner et al., 1998) applied to word embeddings. It computes the minimal cost of transforming one document's words into another's while solving a constrained optimization problem with two constraints. However, WMD has a high computational cost due to its cubic inference complexity.

WCD Rubner et al. (1998) propose a linear complexity loose lower bound of WMD which Kusner et al. (2015) call Word Centroid Distance (WCD). To calculate the distance between documents, WCD first calculates their centroids, i.e. the center or average of their word vectors. Then the Euclidean distance between the centroids of these documents is calculated:

$$WCD(x, y) = \sqrt{\frac{1}{|x|}\sum_{i=1}^{|x|} E(x_i) - \frac{1}{|y|}\sum_{j=1}^{|y|} E(y_j)} \quad (1)$$

In Eq. (1), x and y are two documents compared and E is an embedding function.

RWMD Kusner et al. (2015) also propose the much tighter RWMD, which removes one of the two constraints in the WMD optimization problem. Given the distance of every word in the first document to every word in the other document, RWMD can be calculated with a quadratic complexity.

3.3 Adapters

COMET We incorporate adapters into the training pipeline of the COMET metric to enhance training efficiency by replacing the backbone model from the default pre-trained transformer with its adapter-enabled version.

COMETINHO Furthermore, we apply the same approach to a distillation process described in Rei et al. (2022b). The training procedure for COMETINHO involves creating a large pseudo-labeled dataset with the help of larger COMET models and training a smaller version based on the smaller MiniLM (Wang et al., 2020) pre-trained model (instead of XLM-Roberta-Large (Conneau et al., 2020)) and a smaller estimator layer.

Due to limited hardware resources, we reduce the total training data for COMETINHO. Nevertheless, our goal is not to exactly replicate these models but to investigate whether training metrics can be efficiently obtained without substantial quality loss.

4 Experimental Setup

Following related work, we measure the success of our optimized metrics based on the time needed to calculate them, the memory used, and the storage needed to save the program or related data.

4.1 Evaluation Protocol (disk space, inference time, quality)

For **untrained** metrics, we assess efficiency by measuring runtime, memory usage, and model size and compare these with Pearson's r as a quality measure of the metric (correlation with human assessments).

For the **trainable** COMET metric, we evaluate forward pass and backward pass speeds in tokens per second and memory usage as MB per token. Using relative measures allows us to conduct experiments more efficiently and receive results on the same scale, regardless of batch size and distributed training configuration. The metric's quality is assessed using Kendall τ as a correlation with human evaluations. We choose Kendall τ to make our results more comparable to similar publications for trained metrics such as Rei et al. (2022b).

4.2 Untrained metrics

Runtime To calculate the execution/inference time, we measure the timestamp immediately before starting inference and immediately after ending, then report the difference.

Since the computing speed of a system depends on factors such as hardware and especially the scheduling of tasks by the operating system, runtimes vary from one run to the next, even when all internal variables stay the same. To reduce this variation, we run every experiment at least three times and average the measured runtimes. For comparability between different metrics, we set the batch size of each metric to 1. We also present an ablation study on the impact of different batch sizes in Appendix D. It shows that while the ranking of the models in terms of efficiency might differ with higher batch sizes, our main claim (see below: that TinyBERT provides the best tradeoff between metric quality and efficiency) still holds. For comparability between datasets, we divide the total runtime of a metric by the number of segments.

Memory usage, parameters, and disk space During inference, we measure the peak memory usage of the program. Since the metrics use the transformers library built on PyTorch, we use PyTorch's `memory_stats` array to get the peak usage. We further list the number of parameters in a model and the size needed to save it on a hard drive.

Data We use datasets from WMT15, WMT16, and WMT21, all published by the annual Machine Translation conference WMT. The organizers provide source texts, machine translations, and human references. The texts are from various categories, but we only use the ones from *newstest*. The organizers also publish human assessments, which we correlate with our metrics' output as a quality measure. For WMT15 and WMT16, the human scores are Direct Assessment (DA) (Stanojević et al., 2015; Bojar et al., 2016), while for WMT21, they follow the MQM framework (Lommel et al., 2014).

WMT15 provides scored data in 5 language pairs (4 of which are to-English) with 500 segments each, in a total of 2000 segments. WMT16 provides scored data in 7 language pairs (6 of which are

to-English) with 560 segments each, in a total of 3360 segments.

For WMT21, scored data is available for three language pairs (en-de, en-ru and zh-en). For the only to-English language pair of those, WMT provides scores for 650 segments, per system. To speed up analysis, we reduce the data to the outputs of only five MT systems (*DIDI-NLP*, *Facebook-AI*, *MiSS*, *NiuTrans* and *SMU*), in total 3250 segments. Appendix B provides an overview of language pairs and the number of segments available.

Hardware To get results that are less dependent on our hardware, we consider two different setups: (i) **Virtual Machine on a personal computer**: The first setup is a virtual machine on a personal computer. The VM has an Intel Core i5-10310U CPU (4 cores, 1.70GHz) and 10 GiB RAM. This setup does not have a GPU that can be used for calculations. (ii) **Compute Cluster**: The second setup is the Compute Cluster of a TU Darmstadt Department of Computer Science. For our experiments, we chose an Intel Xeon Gold 5218R CPU (20 cores each, 2.10GHz), with a main memory of 64GiB, 8 Nvidia A100 GPUs with 40GB memory each, and runs CentOS Linux 7 as OS. Using the Slurm Workload Manager (version 20.02.2), we limited the usable CPU cores to 4 and GPUs to 1 during our experiments.

For each experiment, we report the runtime on the CPU (as an average of setup 1 and setup 2) and on the GPU (only from setup 2).

4.3 Trainable metrics

Measuring Training Efficiency To evaluate the impact of different adapter configurations on training efficiency, we measure both model pass speed and memory usage.

The forward pass speed is assessed by recording timestamps immediately before and after the model forward passes, including the computation of the loss function. The backward pass speed is measured by recording timestamps immediately before and after executing backward pass on the model. The difference between these two timestamps is divided by the total number of tokens in the current minibatch to obtain a normalized speed value. For memory usage measurement, we employ PyTorch memory measurement utilities. Prior to the training step, we reset the current memory usage peak and record the new peak immediately after the step. The final memory usage value is obtained by dividing the memory usage peak by the total number of tokens in the current minibatch. Although our primary focus in this section is on training efficiency, since inference is covered in another part of the paper, readers can still gain insights into the dynamics of inference efficiency through the measured forward pass speed. This metric provides a useful indicator of the model's performance during the inference stage.

Adapter Configurations We examine the following adapter configurations and a reference run without adapters.

• *pfeiffer* (Pfeiffer et al., 2020b): A bottleneck adapter layer is placed only after the feedforward block in each transformer layer. The bottleneck consists of down-projection and up-projection with non-linearity in between and an additional residual connection:

$$h \longleftarrow W_{up} \times (W_{down} \times h) + r$$

Here, W_{down} corresponds to a down-projection weight matrix and W_{up} to an up-projection, respectively, and h is an output of the transformer block and r is a residual connection. See Figure 1 for the architecture of the COMET-like model with *pfeiffer* adapters applied.

• *houlsby* (Houlsby et al., 2019): This configuration places the same bottleneck adapter layer both after multi-head attention and the feedforward layer.

• *parallel* (He et al., 2022): This configuration uses a bottleneck adapter placed in parallel to the transformer layer.

• *compacter* (Karimi Mahabadi et al., 2021): Similar to the bottleneck adapter layer, but instead of linear multiplications, it uses *parametrized hypercomplex multiplication layers* (PHM).

• $(IA)^3$ (Liu et al., 2022): This configuration introduces trainable vectors l_W into different parts of the transformer model in a way that augments every matrix-multiplication layer with elementwise multiplication:

$$h \longleftarrow l_W \odot (W \times x)$$

Here, x is an output of a transformer block.

Data For training COMET-based models, we use Direct Assessment (DA) data for the *news* domain from the WMT2020 (Barrault et al., 2020) shared task training dataset. In total, it consists of 230,756 segments, covering 14 language pairs. For

COMETINHO, we follow the approach of Rei et al. (2022b) and use the data provided by them. We take 10M segments out of the original 45M dataset and compute pseudo-scores using the latest available COMET-22 model[2]. We train those models in a reference-based setting. Model inputs consist of the source text, its reference translation as well as a hypothesis (which is a machine translation). The model is optimized to predict a score in a range between 0 and 1, where 1 means that the machine translation of the source text is perfect.

We evaluate the metric's quality using WMT21 (Akhbardeh et al., 2021) newstest datasets, which consists of 27141 segments: 9750 for *zh-en*, 8959 for *en-de* and 8432 for *en-ru*.

Hardware All training-efficiency-related experiments were conducted on Bielefeld's University Compute Cluster. Each node uses an AMD EPYC 7713 64-core Processor, 4 x Nvidia A40, and 512 GB RAM.

5 Results

We structure this section as follows: We report results for **untrained** (1) reference-based metrics (§5.1), (2) reference-free multilingual metrics (§5.2), (3) consider WMD approximations for untrained MoverScore (§5.3) and then (4) we investigate the impact of adapters on training efficiency and metric quality for COMET and COMETINHO **trained** metrics. Due to space constraints, we relegate details to the appendix C and only list the key results as a summary in the main part.

5.1 Reference-Based Metrics

We first consider the "semantic similarity" metrics MoverScore, BERTScore, and BaryScore, then consider BARTScore, which is based on text generation and uses different transformer types and corresponding efficient variants.

5.1.1 BERTScore, MoverScore, BaryScore

BERTScore, MoverScore, and BaryScore use monolingual BERT-based transformers for embedding references and hypotheses. We replace the embedding model with one pruned model, two distilled models, and an early exiting model. The list of the covered models is as follows: $\text{RoBERTa}_{\text{LARGE}}$ — a baseline, the default model for BERTScore; $\text{BERT}_{\text{BASE}}$ — the default model for MoverScore and BaryScore; $\text{BERT}_{\text{TINY}}$, DistilBERT, TinyBERT and $\text{DeeBERT}_{\text{MNLI}}$. For detailed experimental setup and results, please refer to Appendix §C.1.

Key results We visualize the runtime of the inference on a CPU and the quality achieved by the metrics in Figure 2. The fastest model for BERTScore, MoverScore, and BaryScore is $\text{BERT}_{\text{TINY}}$, with up to 41x speedup; however, its quality decreases substantially. TinyBERT achieves a better speedup-quality ratio, with up to 27x speedup while maintaining reasonable quality. Memory measurements are lower for efficient models, with $\text{BERT}_{\text{TINY}}$ using 61x and TinyBERT using 18x less memory than the baseline on BERTScore. DistilBERT has a lower speedup, but the quality drop is less compared to TinyBERT. DeeBERT performs similarly to $\text{BERT}_{\text{BASE}}$ in quality and efficiency. Further, we observe that speedups — no matter on which metric, model, or dataset — are usually not as big on GPU as on CPU, see in Figure 4 in Appendix C.

5.1.2 BARTScore

BARTScore originally uses $\text{BART}_{\text{LARGE}}$ fine-tuned on CNNDM and Parabank2 (Yuan et al., 2021). Optimizations of BART were researched by Shleifer and Rush (2020). The models with the best results from them use a *Shrink and Fine-Tune* approach and were also trained on CNNDM. Apart from the original models, we test $\text{BART}_{\text{BASE}}$ and several distilled versions — dBART-6-6, dBART-12-3, dBART-12-6-t and dBART-12-9-m. Please refer to Appendix §C.2 for detailed setup and results.

Key result $\text{BART}_{\text{BASE}}$ is the fastest and most memory-efficient model but has a substantialquality decline. The distilled version of the BART model, dBART-6-6, achieves a higher correlation compared to our baseline, $\text{BART}_{\text{LARGE}}$, with 1.8x speedup and 1.7x memory efficiency, making it a better choice for quality and efficiency.

5.2 Reference-Free Metrics

XMoverScore (Zhao et al., 2020) and SentSim (Song et al., 2021) are reference-less metrics,[3] i.e., they compare hypotheses directly to source texts. Therefore, both of them use a multilingual embedding model since the source and hypothesis are in different languages in

[2] https://huggingface.co/Unbabel/wmt22-comet-da

[3] When running SentSim, we use the implementation from Belouadi and Eger (2023) since it is better structured and integrates better with our evaluation framework.

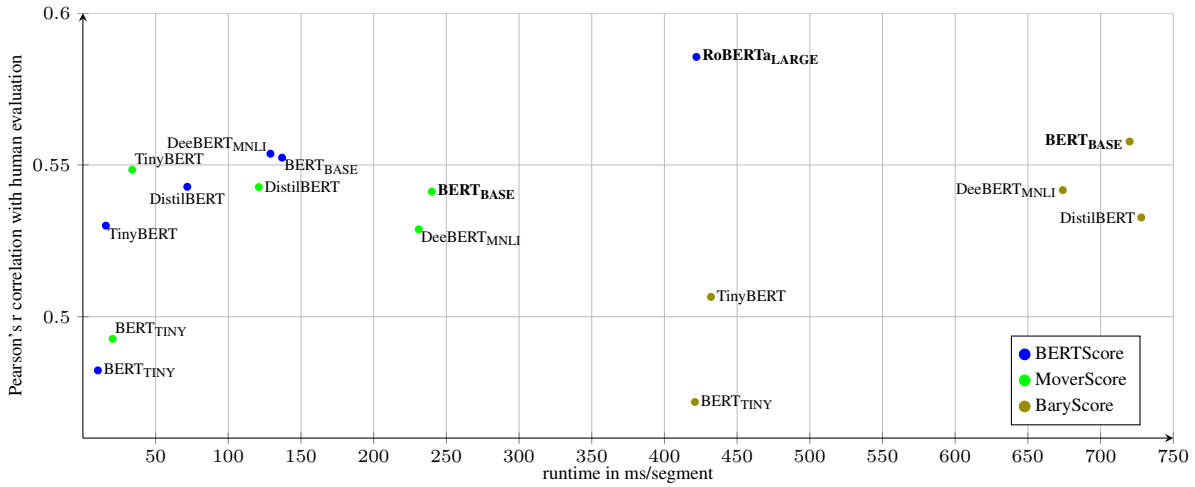

Figure 2: CPU runtime / correlation plot of BERTScore, MoverScore, BaryScore

MT. Zhao et al. (2020) realign multilingual embedding spaces on parallel sentences using fast-align (Dyer et al., 2013). We use WikiMatrix (Schwenk et al., 2021), instead, to train remappings because it contains more languages. We use the original remappings from Zhao et al. (2020) for the baseline. We report Pearson's r with (*remap*) and without remapping (*direct*). XMoverScore further uses a language model for calculating the perplexity of the hypothesis. We also replace this model with a lighter one and measure the efficiency and quality changes. Besides a word level model, SentSim calculates the cosine distance of sentence embeddings (Reimers and Gurevych, 2019, 2020) of the source and hypothesis sentence. We replace this sentence embedding model with a lighter one. We test 4 models from the 'sentence-transformers' model repository: A (XLM-Roberta) — a baseline, B (DistilUSE), C (MiniLM), and D (MPNet). For details on the experiment, see Appendix §C.3.

Key results Both mMiniLMs outperform their baselines in quality and efficiency for XMoverScore and SentSim metrics. mMiniLM$_{12}$ achieves a 0.032 higher correlation than the baseline on XMoverScore (+8.4%). Replacing the language model in XMoverScore with DistilGPT-2 results in a 20% memory reduction and an 8.4% drop in correlation. For SentSim with sentence embedding models, Model C (MiniLM) shows the best combination of efficiency and quality. It achieves a 1.5x faster speed on the CPU, uses 1.4x less memory, and only occupies 43% of the disk space compared to baseline Model A (XLM-R). Quality drop, in this case, is 7.8%.

5.3 WMD

We replace WMD in MoverScore and XMoverScore with more efficient variants to speed up runtime. We implement the two variants WCD and RWMD, which have linear and quadratic complexities respectively. See Appendix §C.4 for more details.

Key Results WMD's efficient variants (RWMD and WCD) perform worse in quality for XMoverScore but see an increased quality for MoverScore when using RWMD on WMT21. Their runtime speedup is not substantial due to the time-consuming embedding calculation.

5.4 Trainable metrics

Config	Mem.↓	Fwd.↑	Bwd.↑	τ ↑
pfeiffer	4.88	5123	**4808**	0.273
parallel	4.97	5128	4525	**0.289**
houlsby	4.87	4607	4036	0.273
compacter	**4.80**	3649	3049	0.269
$(IA)^3$	5.76	**5195**	4712	0.268
no adapters	7.32	6247	2238	0.275
reference	-	-	-	**0.290**

Table 1: Training efficiency of COMET models. **Mem.** is the median memory usage in MB per token, **Fwd.** and **Bwd.** are median values of forward pass and backward pass speed respectively, in tokens per second. τ is the average Kendall τ across languages in the test set. *reference* is the result of applying the latest available COMET-22 model (*Unbabel/wmt22-comet-da*) through official implementation, both released under Apache 2.0 License

For both COMET and COMETINHO, we evalu-

Config	Mem.↓	Fwd.↑	Bwd.↑	τ ↑
pfeiffer	0.770	25499	25774	0.252
parallel	**0.741**	26109	**26113**	**0.252**
houlsby	0.769	23746	21678	0.252
compacter	0.776	18382	15671	0.243
$(IA)^3$	0.997	**27075**	24804	0.248
no adapters	1.012	31836	18941	0.243
reference	-	-	-	**0.241**

Table 2: Training efficiency of COMETINHO models. **Mem.** is the median memory usage in MB per token, **Fwd.** and **Bwd.** are median values of forward pass and backward pass speed respectively, in tokens per second. τ is the average Kendall τ across languages in the test set. *reference* is the result of applying the distilled COMET model *eamt22-cometinho-da* [4] through the official implementation.

ate all 5 adapter configurations (*pfeiffer*, *parallel*, *houlsby*, *compacter*, $(IA)^3$) + 1 control configuration without adapters. Each configuration is tested three times with different random seeds to minimize the impact of random fluctuations. Our total computational budget for these experiments is approx. 888 GPU-hours: 222 hours of compute time $\times$ 4 A40 GPUs used in parallel.

Results for COMET are presented in Table 1. We observe that adapters can improve metric quality compared to standard training, with the parallel configuration showing a higher average Kendall τ correlation and almost matching the reference model on 1/4 training data. Although adapters have a slower forward pass speed, increased backward pass speed compensates for it. Lightweight adapter configurations (*pfeiffer* and *parallel*) have higher backward pass speeds than heavyweight compacter, and these adapters also do not show reduced metric quality compared to them.However, the model with the compacter adapter has the smallest memory footprint of 4.80 MB per token.

For COMETINHO, similar patterns are observed (Table 2). The *parallel* adapters offer the best metric quality and training efficiency, with a 34% memory reduction and 3.7% higher Kendall τ correlation. This model surpasses the reference model with only 22% training data.

6 Analysis & Discussion

We conduct a deeper analysis on reference-based semantic-similarity metrics BERTScore, MoverScore and BaryScore, all of which use the same efficient encoder architectures (we remove RoBERTa-Large from the analysis, as it has only been used with BERTScore). We compute results individually across the three metrics and the three WMT datasets (then report individual results or averages).

Model	Quality	Runtime (CPU)
$\text{BERT}_{\text{BASE}}$	1.00	1.00
$\text{BERT}_{\text{TINY}}$	0.88	8.95
DistilBERT	0.99	1.63
TinyBERT	0.97	5.42
$\text{DeeBERT}_{\text{MNLI}}$	0.96	1.06

Table 3: Quality-inference values.

Which speedups are obtained? Table 3 shows the absolute speedups of encoders and their relative performance deterioration relative to BERT. On a CPU, TinyBERT is 5.4x faster at interference while retaining 97% of the quality of the original metric, which yields the best tradeoff of quality and inference time. $\text{BERT}_{\text{TINY}}$ yields almost 9x faster inference but at the cost of 12 points of performance deterioration.

To illustrate in absolute values, BERTScore with RoBERTa-Large model takes, on average in a GPU environment, 34ms per segment across all three datasets. The same BERTScore with TinyBERT takes 15ms per segment. Considering the practical examples made in the introduction (30k segments $\times$ 5 language pairs $\times$ 50 MT systems), the reference BERTScore would take, as we stated before, approx. 71 hours for a full pass, producing 8kg CO_2-eq of carbon footprint. Our suggested alternative, TinyBERT, would take more than twice as little, 31 hours, producing 3.62kg CO_2-eq. In a CPU environment, the difference becomes even more striking. The reference metric with RoBERTa-Large would take more than 950 hours for a single full pass, while TinyBERT would complete in just under 45 hours. The carbon footprint is 5.4kg CO_2-eq for RoBERTa-Large and 0.26kg CO_2-eq for the TinyBERT-based metric (while maintaining 97% of the reference model's quality).

For those calculations, we assume that a GPU environment (Nvidia A100) has a power draw of 300W under full load, while a CPU environment has a power draw of 15W. However, the small size of TinyBERT would allow practitioners to use lower-tier GPUs, including some mobile GPUs in laptops, which are much more energy-efficient. For

carbon footprint, we take the US 2021 power grid carbon intensity[5] as a reference point.

How stable are the results? We correlate the stability of (normalized) results using Pearson correlation over vectors corresponding to (a) metrics and (b) datasets. Concerning metrics, we observe high Pearson correlations from 0.72 (MoverScore-BaryScore, Quality) to 0.99 (MoverScore-BERTScore, inference time). This means that the LMs perform similarly (in terms of quality, inference time, or both) for each metric. Across datasets, the correlation is high for WMT15-WMT16 (0.91-0.99 for quality, runtime, and average) but considerably lower between WMT21 and the other two (0.20-0.96). This is mainly because quality is not stable: for example, DeeBERT performs badly for WMT21, but quite well for WMT15 and WMT16. We note, however, that the score ranges for WMT21 are low anyway, and DeeBERT absolutely does not perform much lower than the other encoders here.

Which adapters are better? For both COMET and COMETINHO, the 'parallel' adapter configuration consistently outperforms others, achieving high Kendall τ correlation values (0.289 for COMET, 0.254 for COMETINHO) and balancing forward and backward pass speeds. COMETINHO's 'parallel' configuration also exhibits a 34% reduction in memory usage and a remarkable backward pass speed.

Though it remains unclear why adapter-trained models surpass standard models in terms of quality, training curves are often close and sometimes overlapping. The *parallel* adapter model achieves higher Kendall τ values despite reduced trainable parameters, which aligns with recent findings in Nouriborji et al. (2023).

Our experiments utilize reference model hyperparameters, except for batch size and learning rate. Thus, further hyperparameter optimization might produce even higher correlations with human assessments.

7 Conclusion

We have investigated efficient evaluation metrics for natural language generation, particularly MT, in monolingual reference-based and multilingual reference-free versions via three approaches: (i) replacing transformers in metrics by efficient variants, (ii) replacing alignment models in metrics (precisely: Word Mover Distance) with efficient approximations, (iii) training COMET and COMETINHO metrics in a parameter-efficient way with adapters. We have explored multiple types of efficient transformers, finding that TinyBERT shows the best quality-efficiency tradeoff for semantic similarity-based metrics: on average, it retains 97% quality while being more than 5x faster at inference time and having considerably fewer parameters and lower memory consumption. In several cases, we have also identified faster models that yield higher quality at the same time. Finally, we found that efficient alignments on top of transformers do not result in efficiency gains but have adverse effects on quality in 2 out of the 3 datasets we examined.

Our experiments further demonstrate that the 'parallel' adapter configuration consistently outperforms others in efficiency and metric quality for both COMET and COMETINHO. The adapter-trained models achieve faster results, using less memory, and requiring a smaller portion of the reference model's training data, with COMET-sized adapter models being around 30% faster to fully train (3h30m vs. 5h4m). These findings indicate that adapter-based training offers a promising approach for natural language generation tasks, providing optimal memory usage, computational efficiency, and alignment with human assessment.

In future work, we want to combine efficiency with other highly desirable properties of evaluation metrics such as *robustness* (Vu et al., 2022; Chen and Eger, 2023; Rony et al., 2022) and *explainability* (Kaster et al., 2021; Sai et al., 2021; Fomicheva et al., 2021; Leiter et al., 2022) to induce metrics that jointly satisfy these criteria.

Acknowledgements

The Natural Language Learning Group is funded by the BMBF project «Metrics4NLG» and the DFG Heisenberg grant EG 375/5-1.

8 Limitations

In this section, we acknowledge several limitations of our research. First, with respect to WMD Approximations, we observe a surprisingly big drop in quality, on 2 out of 3 datasets, compared to the exact version. Thus, we cannot rule out that we made a mistake in the implementation. Nevertheless, our overarching conclusions remain valid

[5] https://www.eia.gov/tools/faqs/faq.php?id=74&t=11

since the performance improvements achieved by the WMD approximations are negligible in comparison to the time spent calculating the contextualized embeddings. Second, the experiments related to training efficiency with adapters reported in this paper focus primarily on the model's performance on the WMT2021 *newstest* test set, which includes predominantly high-resource language pairs. Consequently, the results obtained in this study may not necessarily extend to other datasets or lower-resource language pairs. Lastly, the experiments with trained metrics were conducted using default adapter configurations and original hyperparameters from respective papers, including those of COMET and COMETINHO. A more comprehensive hyperparameter search could potentially improve metric quality and training efficiency even further. In particular, our models demonstrate superior performance compared to reference COMET and COMETINHO, albeit being trained on smaller datasets, suggesting that either the COMET's & COMETINHO's hyperparameter configurations might not be the best one or that they were over-trained with more data than needed.

References

Farhad Akhbardeh, Arkady Arkhangorodsky, Magdalena Biesialska, Ondřej Bojar, Rajen Chatterjee, Vishrav Chaudhary, Marta R. Costa-jussa, Cristina España-Bonet, Angela Fan, Christian Federmann, Markus Freitag, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Leonie Harter, Kenneth Heafield, Christopher Homan, Matthias Huck, Kwabena Amponsah-Kaakyire, Jungo Kasai, Daniel Khashabi, Kevin Knight, Tom Kocmi, Philipp Koehn, Nicholas Lourie, Christof Monz, Makoto Morishita, Masaaki Nagata, Ajay Nagesh, Toshiaki Nakazawa, Matteo Negri, Santanu Pal, Allahsera Auguste Tapo, Marco Turchi, Valentin Vydrin, and Marcos Zampieri. 2021. Findings of the 2021 conference on machine translation (WMT21). In *Proceedings of the Sixth Conference on Machine Translation*, pages 1–88, Online. Association for Computational Linguistics.

Loïc Barrault, Magdalena Biesialska, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Matthias Huck, Eric Joanis, Tom Kocmi, Philipp Koehn, Chi-kiu Lo, Nikola Ljubešić, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Santanu Pal, Matt Post, and Marcos Zampieri. 2020. Findings of the 2020 conference on machine translation (WMT20). In *Proceedings of the Fifth Conference on Machine Translation*, pages 1–55, Online. Association for Computational Linguistics.

Jonas Belouadi and Steffen Eger. 2023. UScore: An effective approach to fully unsupervised evaluation metrics for machine translation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 358–374, Dubrovnik, Croatia. Association for Computational Linguistics.

Ondřej Bojar, Yvette Graham, Amir Kamran, and Miloš Stanojević. 2016. Results of the WMT16 metrics shared task. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 199–231, Berlin, Germany. Association for Computational Linguistics.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of text generation: A survey. *ArXiv*, abs/2006.14799.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Yanran Chen and Steffen Eger. 2023. Menli: Robust evaluation metrics from natural language inference. *Transactions of the Association for Computational Linguistics*, 11:804–825.

Pierre Colombo, Guillaume Staerman, Chloé Clavel, and Pablo Piantanida. 2021. Automatic text evaluation through the lens of Wasserstein barycenters. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10450–10466, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Marina Fomicheva, Piyawat Lertvittayakumjorn, Wei Zhao, Steffen Eger, and Yang Gao. 2021. The Eval4NLP shared task on explainable quality estimation: Overview and results. In *Proceedings of*

the 2nd Workshop on Evaluation and Comparison of NLP Systems, pages 165–178, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Markus Freitag, Ricardo Rei, Nitika Mathur, Chi-kiu Lo, Craig Stewart, Eleftherios Avramidis, Tom Kocmi, George Foster, Alon Lavie, and André F. T. Martins. 2022. Results of WMT22 metrics shared task: Stop using BLEU – neural metrics are better and more robust. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 46–68, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166*.

Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Hassan Sajjad, Preslav Nakov, Deming Chen, and Marianne Winslett. 2021. Compressing large-scale transformer-based models: A case study on BERT. *Transactions of the Association for Computational Linguistics*, 9:1061–1080.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a Unified View of Parameter-Efficient Transfer Learning. ArXiv:2110.04366 [cs].

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. ArXiv:2106.09685 [cs].

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Moussa Kamal Eddine, Guokan Shang, Antoine Tixier, and Michalis Vazirgiannis. 2022. FrugalScore: Learning cheaper, lighter and faster evaluation metrics for automatic text generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1305–1318, Dublin, Ireland. Association for Computational Linguistics.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems*, 34:1022–1035.

Marvin Kaster, Wei Zhao, and Steffen Eger. 2021. Global explainability of BERT-based evaluation metrics by disentangling along linguistic factors. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8912–8925, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 957–966. JMLR.org.

Christoph Leiter, Piyawat Lertvittayakumjorn, M. Fomicheva, Wei Zhao, Yang Gao, and Steffen Eger. 2022. Towards explainable evaluation metrics for natural language generation. *ArXiv*, abs/2203.11131.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Haokun Liu, Derek Tam, Muqeeth Mohammed, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems*.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. Gpteval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Ro{bert}a: A robustly optimized {bert} pretraining approach.

Arle Lommel, Hans Uszkoreit, and Aljoscha Burchardt. 2014. Multidimensional quality metrics (mqm): A framework for declaring and describing translation quality metrics. *Tradumàtica*, (12):0455–463.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. 2022. UniPELT: A unified framework for parameter-efficient language model tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6253–6264, Dublin, Ireland. Association for Computational Linguistics.

Benjamin Marie, Atsushi Fujita, and Raphael Rubino. 2021. Scientific credibility of machine translation research: A meta-evaluation of 769 papers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7297–7306, Online. Association for Computational Linguistics.

Subhabrata Mukherjee and Ahmed Hassan Awadallah. 2020. XtremeDistil: Multi-stage distillation for massive multilingual models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2221–2234, Online. Association for Computational Linguistics.

Mohammadmahdi Nouriborji, Omid Rohanian, Samaneh Kouchaki, and David A. Clifton. 2023. MiniALBERT: Model distillation via parameter-efficient recursive transformers. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1161–1173, Dubrovnik, Croatia. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Annual Meeting of the Association for Computational Linguistics*.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020): Systems Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Ricardo Rei, José G. C. de Souza, Duarte Alves, Chrysoula Zerva, Ana C Farinha, Taisiya Glushkova, Alon Lavie, Luisa Coheur, and André F. T. Martins. 2022a. COMET-22: Unbabel-IST 2022 Submission for the Metrics Shared Task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 578–585, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Ricardo Rei, Ana C Farinha, José G.C. de Souza, Pedro G. Ramos, André F.T. Martins, Luisa Coheur, and Alon Lavie. 2022b. Searching for COMETINHO: The little metric that could. In *Proceedings of the 23rd Annual Conference of the European Association for Machine Translation*, pages 61–70, Ghent, Belgium. European Association for Machine Translation.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4512–4525, Online. Association for Computational Linguistics.

Md Rashad Al Hasan Rony, Liubov Kovriguina, Debanjan Chaudhuri, Ricardo Usbeck, and Jens Lehmann. 2022. RoMe: A robust metric for evaluating natural language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5645–5657, Dublin, Ireland. Association for Computational Linguistics.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. 1998. A metric for distributions with applications to image databases. *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 59–66.

Ananya B. Sai, Tanay Dixit, Dev Yashpal Sheth, Sreyas Mohan, and Mitesh M. Khapra. 2021. Perturbation CheckLists for evaluating NLG evaluation metrics. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7219–7234, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Holger Schwenk, Vishrav Chaudhary, Shuo Sun, Hongyu Gong, and Francisco Guzmán. 2021. WikiMatrix: Mining 135M parallel sentences in 1620 language pairs from Wikipedia. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1351–1361, Online. Association for Computational Linguistics.

Sam Shleifer and Alexander M. Rush. 2020. Pre-trained summarization distillation. *ArXiv*, abs/2010.13002.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems*, 33:16857–16867.

Yurun Song, Junchen Zhao, and Lucia Specia. 2021. SentSim: Crosslingual semantic evaluation of machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3143–3156, Online. Association for Computational Linguistics.

Miloš Stanojević, Amir Kamran, Philipp Koehn, and Ondřej Bojar. 2015. Results of the WMT15 metrics shared task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 256–273. Association for Computational Linguistics.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4323–4332, Hong Kong, China. Association for Computational Linguistics.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2020. Well-read students learn better: On the importance of pre-training compact models.

Patrick von Platen. 2022. DistilGPT2.

Doan Nam Long Vu, Nafise Sadat Moosavi, and Steffen Eger. 2022. Layer or representation space: What makes BERT-based evaluation metrics robust? In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3401–3411, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. MiniLMv2: Multi-head self-attention relation distillation for compressing pretrained transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2140–2151, Online. Association for Computational Linguistics.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online. Association for Computational Linguistics.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. In *Advances in Neural Information Processing Systems*, volume 34, pages 27263–27277. Curran Associates, Inc.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Wei Zhao, Goran Glavaš, Maxime Peyrard, Yang Gao, Robert West, and Steffen Eger. 2020. On the limitations of cross-lingual encoders as exposed by reference-free machine translation evaluation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1656–1671, Online. Association for Computational Linguistics.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578, Hong Kong, China. Association for Computational Linguistics.

Wei Zhu. 2021. LeeBERT: Learned early exit for BERT with cross-level optimization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2968–2980, Online. Association for Computational Linguistics.

A Related Work

Evaluation metrics Recent years have seen a surge of interest in transformer-based evaluation metrics as these promise higher quality (as measured in correlation with human assessments). Among the first LLM based metrics are BERTScore (Zhang et al., 2020) and MoverScore (Zhao et al., 2019) which model evaluation as a semantic similarity task using contextualized BERT representations. Extensions include BARTScore (Yuan et al., 2021), which reads off probability estimates as metric scores directly from text generation systems, and MENLI (Chen and Eger, 2023), which builds on the paradigm of natural language inference and targets metric robustness. A distinction of evaluation metrics is whether they use human references (in *reference-based* metrics) or do not use them (in *reference-free* metrics), where the latter are less costly. In this work, we consider XMoverScore (Zhao et al., 2020) and SentSim (Song et al., 2021) as reference-free metrics and BARTScore, BERTScore, MoverScore and BaryScore (Colombo et al., 2021) as reference-based metrics. In MT, the challenge for reference-free metrics is cross-lingual representation spaces, for which there are also different efficient transformer variants. Another distinction of metrics is whether they are trained or untrained. **Trainable/trained metrics** such as COMET (Rei et al., 2022a), which are particularly popular in the MT community, address the task of NLG evaluation by directly fine-tuning models on human-sourced annotation scores. COMET is trained on direct assessment scores given by human annotators to translations produced by various MT systems. As a result, COMET outperforms untrained metrics in terms of correlation with human assessment. However, the training process can be expensive due to increased model and data size.[6]

Efficiency is a core issue of modern deep learning systems, which have become bigger and bigger in a quest for better performance, leading to environmental concerns and increasing inequality/exclusion. There are many approaches for obtaining more efficient models, especially in the context of computation-heavy transformers.

Knowledge Distillation involves (i) a *student* with pruned layers, embedding size or attention heads or even with layers replaced by alternative simpler network architectures and (ii) a *teacher* usually of the same architecture, but larger. Then the student is trained with outputs from the teacher (Hinton et al., 2015; Ganesh et al., 2021). While data for traditional training often is scarce, in knowledge distillation, the teacher *generates* new data for training the student (Hinton et al., 2015). Knowledge distillation thus usually leads to a better quality for the student model than traditional training because it benefits from the knowledge of the larger model. For example, DistilBERT (Sanh et al., 2019) consists of only 6 layers, which is half the amount of $\text{BERT}_{\text{BASE}}$, but with reported small reductions of model quality. TinyBERT (Jiao et al., 2020) introduces an improved method for knowledge distillation in which the student is trained with outputs (as DistilBERT does) and intermediate results.

Kamal Eddine et al. (2022) apply knowledge distillation to induce an evaluation metric which they call FrugalScore. They employ knowledge distillation on a set of pre-trained miniature BERT models, which were fine-tuned using a synthetic dataset created with full-size models employing BERTScore and MoverScore metrics. In contrast to their approach, we much more comprehensively analyze efficiency in evaluation metrics. Especially we consider more datasets and metrics, explore different evaluation environments (CPU-only and GPU), and improve training efficiency through adapters.

Dynamic Inference Acceleration: Sun et al. (2019); Xin et al. (2020); Zhu (2021) compare intermediate results after each encoder layer, after which they add an early exit ramp. If the change of the representation from one layer to the next is lower than a threshold, they assume later layers will improve even less and skip them by taking the early exit ramp. The final output for that sample will then be the representation from that layer.

Adapters is an approach that addresses the problem of training-time efficiency for fine-tuning transformer models. Popularized by Pfeiffer et al. (2020a); Houlsby et al. (2019) in the NLP community, adapters propose to freeze all parameters of the pre-trained transformer model and instead train adapters — small intermediate layers added into the model graph. This reduces required mem-

[6] A different approach to a text evaluation is presented in the recent papers of Fu et al. (2023); Liu et al. (2023), which form a novel class of LLM based metrics. Those metrics rely on LLM predictions to assess different aspects of the text, with GPTScore (Fu et al., 2023) relying on token probabilities, while GPTEval (Liu et al., 2023) uses Chain-of-Thought to prompt the model to generate scores.

ory usage because the gradients are computed only for a small number of parameters, though it is still required to keep the original model in memory, and can also lead to improvements in training speed. There are multiple ways to apply an adapter to the model. Simpler adapters (Houlsby et al., 2019; Pfeiffer et al., 2020b; He et al., 2022) offer a straightforward architecture with minimal modifications, focusing on training time and memory efficiency, while more complex ones (Hu et al., 2021; Karimi Mahabadi et al., 2021; Liu et al., 2022) provide the potential for better performance improvements at the cost of increased architecture complexity and resource consumption. For instance, the recently introduced $(IA)^3$ (Liu et al., 2022) adapter architecture is specifically designed to maximize fine-tuned accuracy after a few training steps. Another example is the method called UniPELT (Mao et al., 2022), which combines prefix-tuning, low-rank adaptation (LoRA), and adapter layers.

B WMT Data

pair	WMT15	WMT16	WMT21
cs-en	500	560	
de-en	500	560	
fi-en	500	560	
ro-en		560	
ru-en	500	560	
tr-en		560	
zh-en			3250
total	2000	3360	3250

Table 4: Overview of the amount of segments per language pair in each dataset.

C Experiment details

C.1 BERTScore, MoverScore, BaryScore

Setup Although BERTScore was built using various models, for comparing sentences in English (as both our references and hypotheses are), the authors suggest using $\text{RoBERTa}_{\text{LARGE}}$ (Zhang et al., 2020). We only use it on BERTScore, since it was too slow with MoverScore and BaryScore. Its configuration is: $L = 24, H = 1024, A = 16$ (Liu et al., 2020), where L is the number of layers, H the size of each layer and A stands for the attention heads. **$\text{BERT}_{\text{BASE}}$** is the original model of MoverScore and BaryScore (Zhao et al., 2019; Colombo et al., 2021) and also a possible optimization for BERTScore. It has $L = 12$, $H = 768$ and $A = 12$ (Devlin et al., 2019). **$\text{BERT}_{\text{TINY}}$** is the smallest variant of BERT (Turc et al., 2020) and was trained in the traditional way (directly on data, no Knowledge Distillation). It has $L = 2$, $H = 128$, $A = 2$. **DistilBERT** is a distillation of $\text{BERT}_{\text{BASE}}$. It has the same hidden dimensions of 768 and 12 attention heads, but only $L = 6$. Also, they optimized the final output layers (Sanh et al., 2019). **TinyBERT** is another distillation of $\text{BERT}_{\text{BASE}}$ with a more robust training method. It has $L = 4$, $H = 312$, and $A = 12$ (Jiao et al., 2020). **$\text{DeeBERT}_{\text{MNLI}}$** is an early exiting version of $\text{BERT}_{\text{BASE}}$. It has the same structure, but after each encoder layer there is one added classification layer, that can be used as an off-ramp, to stop inference at intermediate states (Xin et al., 2020).

Results We observe a speedup of the runtime that coarsely correlates with the size of the model. The figure shows that the fastest model on each of the metrics is $\text{BERT}_{\text{TINY}}$, which is up to 41x faster than the baseline (on BERTScore), but its quality also decreases by over 10 points correlation for BERTScore. A better speedup-quality ratio achieves TinyBERT: The quality decreases by less than 6 points across all metrics (improves even for MoverScore). Furthermore, it is still up to 27x faster than the baseline (on BERTScore). Memory measurements show similar behavior to runtimes, and they coarsely correlate with model size. $\text{BERT}_{\text{TINY}}$ uses 61x and TinyBERT 18x less memory than the baseline on BERTScore. Compared to TinyBERT, DistilBERT shows lower speedup and memory saving, but also a lower quality decrease (on all three metrics approximately half the decrease of TinyBERT). In our experiments, DeeBERT behaves very similar to $\text{BERT}_{\text{BASE}}$, both in quality and efficiency.

C.2 BARTScore

Setup **$\text{BART}_{\text{LARGE}}$ Para** is the original model used in BARTScore, fine-tuned on the Parabank2 dataset. It consists of 12 encoder and 12 decoder layers and has a hidden size of $H = 1024$ (Lewis et al., 2020). **$\text{BART}_{\text{LARGE}}$ CNN** uses the same architecture, but after pre-training on CNNDM, it was not fine-tuned on Parabank2. We run experiments on $\text{BART}_{\text{LARGE}}$ CNN and used these results as a baseline for fair comparisons to the other models. **$\text{BART}_{\text{BASE}}$** was proposed by Lewis

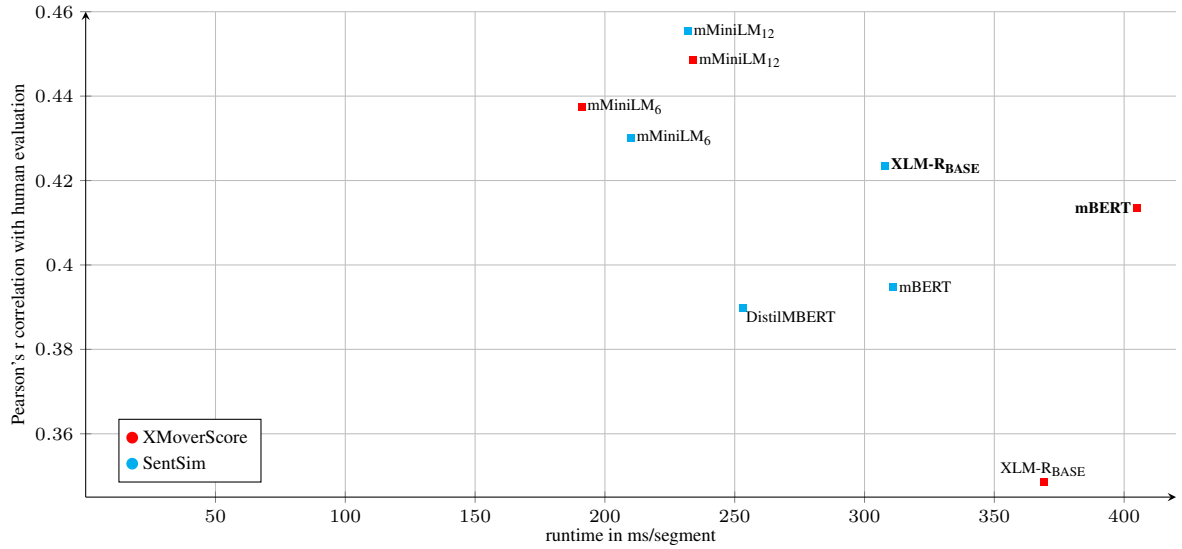

Figure 3: CPU runtime/correlation plot of XMoverScore and SentSim

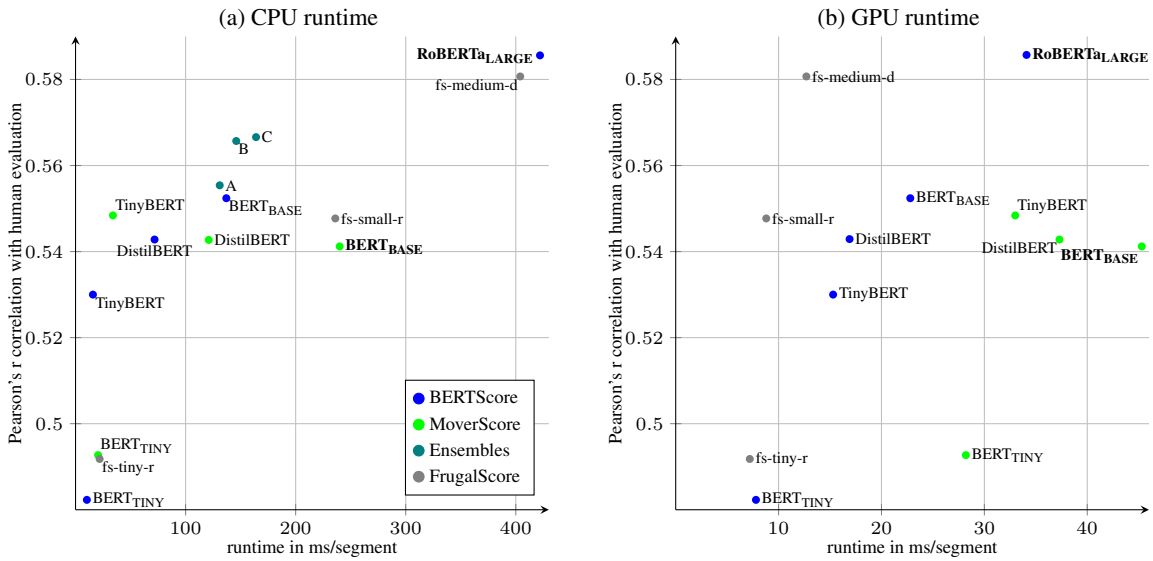

Figure 4: Selected metrics in runtime / correlation plots on a CPU and GPU.

et al. (2020) and differs from $\text{BART}_{\text{LARGE}}$ in having 6 encoder and decoder layers instead of 12 and $H = 768$. **dBART-6-6** is a distilled BART version with 6 encoder layers and 6 decoder layers. It was shrunk from $\text{BART}_{\text{LARGE}}$ and therefore has $H = 1024$ (Shleifer and Rush, 2020). **dBART-12-3** is a distilled BART version with 12 encoder layers and 3 decoder layers. It was shrunk from $\text{BART}_{\text{LARGE}}$ and also has $H = 1024$ (Shleifer and Rush, 2020). **dBART-12-6-t** has 12 encoder layers and 6 decoder layers and was trained on WikiSQL dataset. **dBART-12-9-m** has 12 encoder layers and 9 decoder layers, with $H = 1024$.

Results We observe differences among the baselines: $\text{BART}_{\text{LARGE}}$ Para uses almost twice the memory of $\text{BART}_{\text{LARGE}}$ CNN. The fastest model is again the smallest model – $\text{BART}_{\text{BASE}}$ – with speedups of 2.7x and 1.8x compared to $\text{BART}_{\text{LARGE}}$ CNN. $\text{BART}_{\text{BASE}}$ is also the most memory-efficient model: with the usage of 663MB, it needs 2.6x less than $\text{BART}_{\text{LARGE}}$ CNN. Despite being very efficient, $\text{BART}_{\text{BASE}}$'s quality declines too much, with a correlation coefficient 12 points lower than the baseline (-25%). Concerning quality, dBART-6-6 even gets a higher correlation than $\text{BART}_{\text{LARGE}}$ CNN by 0.02 (+3.6%). It brings a speedup of 1.8x (CPU) and 1.7x (GPU) and is 1.7x more memory efficient. dBART-12-3, dBART-12-6-t and dBART-12-9-m do not achieve competitive quality, also their acceleration over $\text{BART}_{\text{LARGE}}$ CNN is only moderate.

C.3 Reference-free metrics

Setup To implement efficient reference-less metrics, we explore the following multilingual embedding models: **mBERT** is the original model of XMoverScore used by (Zhao et al., 2020). It has $L = 12$, $H = 768$, and $A = 12$. It was trained on 104 languages (Devlin et al., 2019). **XLM-R_{BASE}** is the original model of SentSim used by Song et al. (2021). It has $L = 12$, $H = 768$ and $A = 12$ and was trained on data in 100 languages (Conneau et al., 2020). **DistilMBERT** is a distillation of mBERT (Sanh et al., 2019). It has the same dimensions and attention heads, but only $L = 6$ layers and the final output layers were stripped. These missing output layers are what makes this model incompatible to XMoverScore. Thus, we only use this model for SentSim. **XtremeDistil** is another distillation of mBERT (Mukherjee and Hassan Awadallah, 2020). The model, called *TinyMBERT* in the first version of the paper, has $L = 6$, $H = 256$ and $A = 12$. **mMiniLM_6** is a distillation of XLM-R_{LARGE} with $L = 6$, $H = 384$, and $A = 12$. **mMiniLM_{12}** is a distillation of XLM-R_{LARGE} with $L = 12$ (Wang et al., 2021). **For XMoverScore with language models** the two compared models are: **GPT-2** (Radford et al., 2019) is the original model used by XMoverScore (**?**). **DistilGPT-2** is a distillation of GPT-2 (von Platen, 2022). **For SenSim with sentence embeddings** we replace the original sentence embedding model with a lighter one. We try 3 other models from the SBERT framework. For reasons of clarity, we abbreviate the names to a letter from the SBERT site: **A: xlm-r-bert-base-nli-stsb-mean-tokens**, the original model used by (Song et al., 2021). **B: distiluse-base-multilingual-cased-v2** - a DistilBERT-based model which was fine-tuned on synthetic data created with Universal Sentence Encoder (Cer et al., 2018), **C: paraphrase-multilingual-MiniLM-L12-v2** - based on MiniLM model with 12 layers, and **D: paraphrase-multilingual-mpnet-base-v2** - which is based on MPNet model (Song et al., 2020).

Results We present the results in Figure 3. Both mMiniLMs outperform their baselines in quality and efficiency. The 6-layer version is up to 2.1x faster (on CPU) than the baseline and has a 0.024 higher correlation for XMoverScore (+5.8%). Even higher is the quality improvement with mMiniLM_{12}: it achieves a 0.032 higher correlation than the baseline on XMoverScore (+8.4%). Both models also show quality improvements on SentSim. The space they occupy on a disk is $1/5$ of XLM-R_{BASE} and $1/3$ of mBERT. Although the needed memory is a lot higher than the disk space, with up to 1,594MB (mMiniLM_{12} on SentSim), the models still need 1.2x (SentSim) and 1.4x (XMoverScore) less inference time than the baselines. DistilMBERT on SentSim also shows speedups of 1.2x on both CPU and GPU but has a 0.034 lower Pearson's r than XLM-R_{BASE}. XtremeDistil, despite bringing some memory efficiency and a big saving on disk space, has a quality that is too bad on both metrics. The remappings on XMoverScore do not bring any difference for mBERT and XLM-R_{BASE}. For the mMiniLMs, we observe a quality increase of 1.6 points (6 layers) and 1.3 points (12 layers) correlation compared to using no remapping. Only for XtremeDistil do we see a real difference: using the remap-

pings improves correlation by approximately 0.122 (+110%) compared to using the model directly. For XMoverScore with language models we can not see any change in runtime on a CPU, but observe a speedup of 1.3x on a GPU. We also see a lower memory usage of 317MB (20%) and a lower disk-space of a third. The quality drops by 3.5 points. For SentSim with sentence embedding models we observe a slight speedup of Model B of 1.2x (CPU) and 1.3x (GPU), but also a decrease of quality of 5.5 points. Model C shows only small decrease of quality of 1.7 points, but is very efficient: it runs in 1.5x faster speed on CPU and 1.1x faster on GPU, saves memory (1.4x less) and also only occupies 43% of the disk-space. Model D achieves a slightly higher correlation (+1.2 points) and has the same size as the baseline (comparable speed and memory).

C.4 WMD

Results We see a drop in quality from a correlation of 0.54 on average over all three datasets to 0.43 and 0.39, but surprisingly, we see a substantial increase of quality when using rwmd on WMT21. On the other hand, the quality of XMoverScore, as indicated by the correlation with human scores, declines when using these more efficient variants. wcd achieves correlations approximately 10 percent lower than wmd. As on MoverScore, rwmd's correlations drop approximately 30 percent for WMT15 and WMT16. For WMT21, we again observe a very high correlation with rwmd.

Runtime For neither MoverScore nor XMoverScore do we observe a substantial speedup while using wcd or rwmd instead of wmd. Thus, we investigate the time consumption of each calculation step in more detail.

Step	**WMD**	**WCD**	**RWMD**
get BERT embeddings	285.499	287.915	291.122
calculate distance matrix	0.829	0.005	0.782
calculate distance	5.602	0.616	0.449

Table 5: Runtime (in ms) of each step of MoverScore for various distance functions using its original model $\text{BERT}_{\text{BASE}}$.

In Table 5, we can see that the calculation of wcd and rwmd is substantially faster than wmd — 9x and 12x on MoverScore and 20x and 27x on XMoverScore. But it also shows that the calculation of the embeddings (and of the perplexity) takes a much longer time than the calculation of the distance. No matter how fast the distance can be calculated, the speedup will be eaten up by the variance of the calculation time for the embeddings (and perplexity).

D Impact of different batch sizes

In order to examine the effects of different batch sizes, we conducted reduced experiments with a smaller number of models, testing only on BERTScore and WMT15 dataset. See Table 6 for a list of hyperparameters. As before, we report averaged results of 3 runs to account for fluctuations.

Hyperparameter	Setting
Model	$\text{BERT}_{\text{BASE}}$, $\text{BERT}_{\text{TINY}}$, TinyBERT, DistilBERT
Environment	CPU and GPU cluster nodes
Batch size	1, 4, 16, 64

Table 6: Hypeparameter settings for batch-size ablation study.

Model	1	4	16	64	r
$\text{BERT}_{\text{BASE}}$	.084	.021	.010	.009	.729
DistilBERT	.032	.012	.006	.005	.709
TinyBERT	.015	.015	.003	.002	.707
$\text{BERT}_{\text{TINY}}$	.005	.006	.005	.002	.635

Table 7: Runtime duration (seconds per segment) for BERTScore with given models and variable batch size in CPU environment. r stands for Pearson's correlation with human judgement and provided for brevity.

Model	1	4	16	64	r
$\text{BERT}_{\text{BASE}}$	.021	.006	.003	.002	.729
DistilBERT	.013	.004	.002	.002	.709
TinyBERT	.011	.004	.002	.001	.707
$\text{BERT}_{\text{TINY}}$	.007	.003	.002	.002	.635

Table 8: Runtime duration for BERTScore with given models and variable batch size in GPU environment. r stands for Pearson's correlation with human judgement.

According to the results in Tables 7 and 8 there is a notable change in the model's relative efficiency compared to each other with change of the batch size. nevertheless, the best tradeoff between metric's quality and efficiency is still provided by TinyBERT model. It is also worth noting that in the case of the GPU environment, we observe faster

saturation of efficiency gains between models of different sizes. On higher batch sizes, they perform around the same. However, switching to larger batch sizes leads to progressively higher memory consumption.