Efficient Metrics

Dwarak Vittal, Leon Wempe, Paul Ranly and Sewin Tariverdian
Technical University of Darmstadt
{dwarak.adugodi_vittal, leonjulian.wempe,
paulmoritz.ranly, sewin.tariverdian}@stud.tu-darmstadt.de

Abstract

With the rise of Transformer models and especially BERT (Devlin et al., 2018), many current evaluation metrics for natural language generation i.e. machine translation or summarization make use of these methods to achieve state-of-the-art scores. In this work we replace the most widely used Transformer-models with different variants to produce better or similar results in less time or with lower memory usage. We show that metrics utilizing BERT variants can show even better results in Pearson correlation while also providing opportunities to reduce the time- or memory overhead.

1 Introduction

In Natural Language Generation (NLG), texts are automatically produced to resemble a human author. This can be for instance an automatic translation from another language, text generation, answering of questions, or a summary to an already existing text. To be able to rate a generated text, it needs to be evaluated. This can be done by human evaluators or automated by a machine. An automatic evaluation by a machine provides advantages such as superior speed and fewer human resources needed. Furthermore, if one wants to evaluate a NLG-system to improve the results by for example changing its parameters, a manual evaluation has to be done completely from scratch, taking too much time for regular testing of the system (Sai et al., 2020). Still, it is difficult for a machine to match the human evaluation, which is the desired outcome to produce human-like written texts. To make use of the advantages of an automatic text evaluation and give consistent results, multiple evaluation metrics with different approaches were proposed over the years.

Some of these metrics include BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005) and ROUGE (Lin, 2004). In general, these metrics work on *n-gram* matches (word-overlappings) between a reference solution and the generated machine candidate sentence. This approach in evaluation was soon found to be not correlating well with human judgement, as for example Zhang et al. (2004), Callison-Burch et al. (2006) and Reiter and Belz (2009) showed in their works. Nevertheless, these automatic metrics were commonly used for many years (Gkatzia and Mahamood, 2015).

With the rise of deep neural networks in 2014, many new metrics making use of the new characteristics were proposed. This includes the use of word embeddings (Mikolov et al., 2013) in general (Forgues et al., 2014), contextual word embeddings (Mathur et al., 2019) or deep neural nets and contextualized annotated data (Lowe et al., 2017). The proposal of the Transformer model by Vaswani et al. (2017) led to a new focus of recent metrics, making use of its attention module. This provides a mechanism of weighting tokens according to their importance in the sequence. With this, pre-trained Transformer-models such as BERT (Devlin et al., 2018) could be used in novel metrics, for example BERTSCORE (Zhang et al., 2019), MOVERSCORE (Zhao et al., 2019) or SUPERT (Gao et al., 2020). The attention mechanism of the Transformer helps these metrics to focus on more indicative but maybe rarer words in the input text and thus shows a better correlation with human judgment (Clinciu et al., 2021).

Even more importantly, BERT models are pre-trained on large corpora and can thus be used without further training and still provide competing results (Compare e.g. (Baruah et al., 2020)). The Mask Language Model on which BERT works, provides context for the model not only in the scope of the previous token but surrounding tokens in general (As described e.g. in (Nozza et al., 2020)). This feature further helps the model to grasp the

context.

Generally, automatic evaluation metrics can be separated into two different ways of working, being *reference-based* and *reference-free*. Reference-based metrics, such as the aforementioned BERTSCORE, MOVERSCORE and also BLEURT (Sellam et al., 2020) and COMET (Rei et al., 2020) use a reference solution to improve on their result, building a supervised learning environment. Reference-free metrics, like SUPERT or XMOVERSCORE (Zhao et al., 2020a) on the other hand do not use any reference solutions in their evaluation, which currently makes it more difficult for the metric to achieve human-like judgement, but also does not need previously referenced data.

Since BERT has been been introduced, many new models were developed, such as ROBERTA (Liu et al., 2019) training with different hyperparameters and thus achieving slightly better scores on some tasks, ALBERT (Lan et al., 2019), which is a lite version of BERT with fewer parameters but competing scores, or DISTILBERT (Sanh et al., 2019), a faster and smaller model compared to BERT.[1] These models propose various improvements over the base model by being either smaller, faster or better performing.

In this work, we want to investigate possible advantages of using transformer-based automatic NLG evaluation metrics with selected improved Transformer models to achieve higher human correlation scores, a faster evaluation, or a lower memory footprint while evaluating. To provide more efficient metrics a combination of these improvements in a single model is our desired goal. This would benefit the metrics and possibly improves their results or footprint enough for a new state-of-the-art.

2 Related work

In this section, we briefly describe BERT and its variants and the different metrics we use.

2.1 BERT and its variations

BERT stands for Bidirectional Encoder Representations from Transformers and is a language representation model presented by Devlin et al. (2018). Like the name suggests it learns bidirectional representations from unlabeled text. After the pretraining, the model can be trained for various tasks in the natural language processing field with just minor adjustments. Its strength comes from its conceptual simplicity and it is the empiric power to set new benchmarks in various natural language processing tasks. (Devlin et al., 2018)

Thanks to the success of BERT other scientists have experimented with BERT and have created different variations of it. For example, (Liu et al., 2019) presents RoBERTa which is based on the idea, the pretraining for BERT could be better optimized compared to the originally presented method in (Devlin et al., 2018). Other examples are ALBERT and DistilBERT which are presented by (Lan et al., 2019) and (Sanh et al., 2019). These two BERT variations are more lightweight and trainable with fewer resources compared to the original BERT. There are still other BERT variations that are not mentioned here, but the mentioned ones should provide an idea of how scientists are researching with BERT and how they are trying to achieve better models compared to the performance and usability of the original BERT.

For the multilingual task we compared four different transformers in their performances (”mBERT”, ”XLM-Roberta-base”, ”DistilmBERT”, ”TinymBERT”). mBERT is supposed to be our baseline and bring scores from different transformer into comparison. Thus XLM-Roberta-base does not claim to be a efficient variant of mBERT, it helps to bring the results in to context. DistilmBERT and TinymBERT(Jiao et al., 2019) claim to be smaller and more efficient variants of mBERT, while still achieving similiar performances. Most of the used variations are online available as pretrained models[1]. For TinyMBERT the model had to be downloaded locally to be used. We used the same model as the authors of (Jiao et al., 2021) suggested for a multilingual MT task. The model has 6 layers and 768 hidden layers [2].

2.2 Efficiency

The main goal of more efficient Transformers as described in (Tay et al., 2020) is to decrease the quadratic complexity of the self-attention mechanism and thus make this dense operation more sparse. This can be achieved by several techniques, e.g. fixing the model’s field of view to predefined patterns, approximating the self-attention matrix

[1] A more complete list of (pre-trained) models based on Transformers can be found at https://huggingface.co/models

[2] https://github.com/huawei-noah/Pretrained-Language-Model/tree/master/TinyBERT

or combining distinct patterns. Other methods that do not focus on the self-attention include weight sharing between layers to decrease the size of the model (for instance the RNN-inspired approach by Dehghani et al. (2018)) or Neural Architecture Search, where more efficient Transformers are searched by removing redundant operations of base-Transformers (e.g. Neural Architecture Transformer (Guo et al., 2019)).

One example of an efficient Transformer is the mentioned ALBERT, where parameters are shared across layers to decrease the model's size and improve its parameter efficiency among other techniques.

Another different approach taken by some of the utilized models is Knowledge Distillation as presented by Hinton et al. (2015). Used for instance in DistilBERT and TinyMBERT, distillation describes the process of a knowledge-transfer between an ensemble of models - in literature often called the teacher - and a more compact model, the student. The goal of the student is to mimic the behavior of the teacher so that their performances are similar but the footprint, be it time-, size- or memory-wise, is much smaller for the student. In this way the student is able to adapt properties from the teacher that it would not have learned in its own compact structure.

Doing this distillation benefits the student because it is not just a parameter reduction but a remapping onto a smaller model. As an example the student DistilBERT reduces its number of layers compared to the teacher BERT by a factor of 2 and optimizes operations inside the architecture while also removing some components. By that the computing efficiency is largely impacted. Because the student acts similar to its teacher, the performance of DistilBERT is almost as good as for BERT (compare (Sanh et al., 2019)) while its compactness makes it train and work faster and much more memory-efficient in usage.

TinyMBERT benefits from a *two-stage learning framework*. First the model is trained on a general level by transferring the linguistic knowledge from teacher model, in this case BERT, to a smaller student model. In the next step the issue of overparametrization is approached by fine-tuning BERT on a specific task and teaching its configuration to the student model TinyMBERT.

2.3 BERTScore

BERTScore is a supervised translation evaluation metric and was first presented by Zhang et al. (2019). It makes use of contextual embeddings in BERT (Devlin et al., 2018) to represent word-tokens, which are then compared to the given reference via cosine similarity. The maximum similarity values can then be optionally weighted according to their importance. Figure 1 gives a visual description of the different computation steps in BERTScore.

The results from their research certify BERTScore state-of-the-art results at the time of publication, outscoring different kinds of common metrics, such as *n-gram matching*, *edit-distance-based* or *learned* metrics. After the release of MOVERScore (see 2.4) BERTScore was further improved using MOVERScore's various improvements.

Importance weighting in BERTScore helps focusing on rarer but also more indicative words. It has been previously shown that sentence similarity is highly influenced by indicative words, but those occur much less frequently in a text (for instance Vedantam et al. (2015) use importance weighting for CIDEr). To utilize this knowledge, BERTScore uses the inverse document frequency (idf) to assign a idf score to each word in a sequence.

2.4 MoverScore

MoverScore is built upon the idea of Transformer models and can be used to evaluate different tasks like machine translation, summarization, (response) text generation or image captioning. It was first presented by Zhao et al. (2019) and makes use of the contextualized embeddings BERT and its variations bring[3]. MoverScore also utilizes Word Mover's Distance (Kusner et al., 2015) with soft alignments.

As shown in Figure 2 the hard alignments that e.g. BERTScore uses for calculating its scores focus on one semantically similar word in the other sequence each, while soft alignments between words allow for different words to map on one word in the target sequence or one word to map on many. This makes MoverScore more flexible as it can be seen as a generalization of the distance function used in BERTScore (compare Appendix

[3]In the paper also static embeddings are considered but yield worse results. For this project we only focus on contextualized embeddings

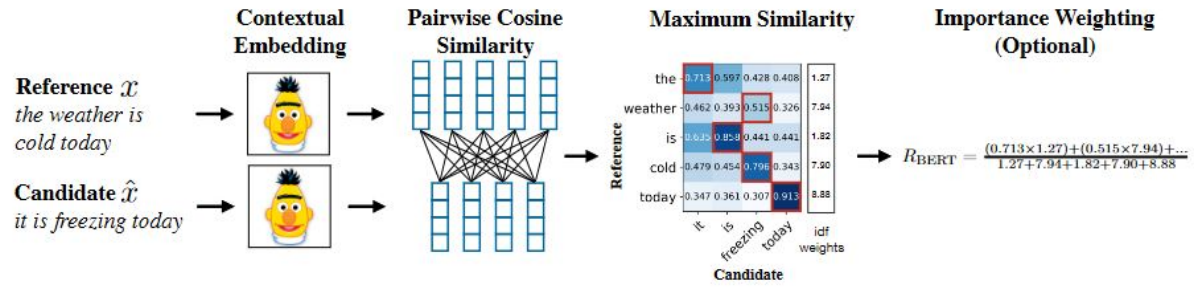

Figure 1: Visualization of the computation of BERTScore from Devlin et al. (2018) between a reference sentence and a candidate translation. The optional importance weighting (see 2.3) is included as the last step in this visualization.

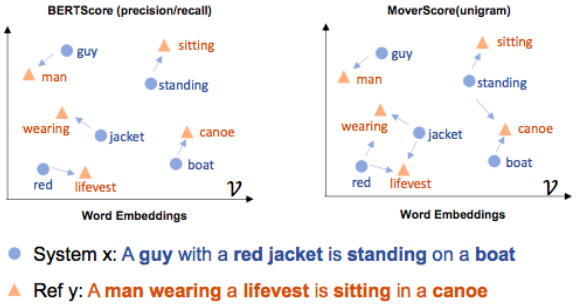

Figure 2: The difference between BERTScore- and MoverScore-alignments. BERTScore uses contextual embeddings with hard alignment, MoverScore relies on soft alignments. Graphic as shown in (Zhao et al., 2019).

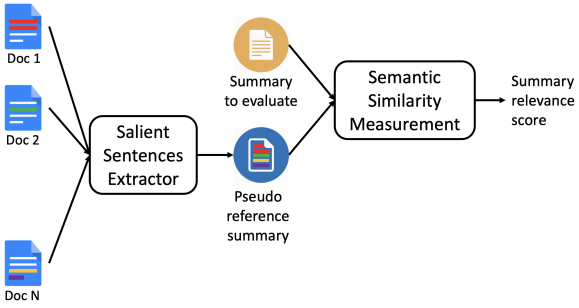

Figure 3: Visual description of SUPERT from Gao et al. (2020)

A in (Zhao et al., 2019)). Because of this, MoverScore can focus much more on the semantics of sequences to compare and not only their structure.

2.5 XMoverScore

XMoverScore is a cross-lingual extension of MoverScore and a reference-free metric for evaluating machine translation. The metric targets especially the issue of "translationese" e.g. word-by-word translation and tries to evaluate the translation based on the semantic distance. The paper (Zhao et al., 2020b) also states that this method does not outperform traditional reference-based metrics like BLEU yet.

2.6 SUPERT

SUPERT (Summarization evaluation with Pseudo references and BERT) is an unsupervised multi-document summarization evaluation metric presented by Gao et al. (2020). It rates the quality of the given summaries by creating a *pseudo reference summary* and comparing its similarity with the summaries, see also Figure 3.

Compared to other state-of-the-art unsupervised evaluation metrics the paper describes SUPERT as superior regarding the correlation with human ratings. Furthermore, the paper presents a way to use SUPERT as a reward function for a neural-based *reinforcement learning summarizer* to have better results compared to other available state-of-the-art unsupervised summarizers.

2.7 COMET

COMET (Rei et al., 2020) is a framework for training metrics for multilingual machine translation models. COMET focuses on the problem that most metrics rate one translation hypothesis higher than another when there are two possible but completely correct ones. To overcome this, COMET uses two models, an estimator model that produces a quality score of the translation and a translation ranking model that reduces the distance between similar correct translation hypotheses. With this strategy, COMET gives state-of-the-art results at the time of publication.

3 Data

For evaluating the multilingual Machine Translation task-metrics we use the WMT machine

translation datasets, i.e. from WMT17[4] and WMT19[5]. The data provides a source in the original language, a generated translation, and a reference translation for evaluation. Besides the data provides a human- and BLEU-score for further evaluation. Two examples are given below.

Source	Translation	Reference	Human Score	BleuScore
Krankheits -fall: Wann bezahlt der Ver- anstalter	disease case : when pay- ing the organiser	when do organis- ers pay if illness strikes ?	-1.7874	0.1531
Coe holte sich dreimal einen Nach- schlag.	coe re- peated three times as a reference .	mr. coe went back for seconds , and thirds .	-1.3974	0.1382

For evaluating SUPERT as an unsupervised multi-document summarization evaluation metric it would normally be obvious to use the two multi-document summarization datasets from the Text Analysis Conference [6] (TAC) shared tasks (TAC'08 and TAC'09) since they are also used in (Gao et al., 2020). Unfortunately the datasets are only available after submitting the required user agreement forms directly from TAC [7] [8]. Since getting approval from TAC for the agreements form would be beyond the scope of this project the provided dataset which comes with the code for SUPERT is used for the experiments with it. The given example is from DUC'09, topic 0939-A [9].

4 Methods

Our approach is to replace BERT in BERTScore, MoverScore, XMoverScore, and SUPERT with other BERT variants. Then we compare the results in terms of general score, time, and space complexity.

5 Experiments

To compare the different BERT models with each other, we evaluate their scoring results (e.g. by the Pearson Correlation with human results), the time taken for the execution of the model, and the maximum memory usage during execution. For the experiments, we show the relative execution time and memory usage in comparison to the base BERT model for the sake of consistency, even though some metrics use different models as default. The default models and resulting consequences for the interpretation of the results will be stated in each paragraph if applicable.
Note that for our experiments we used the base-versions of all models for better comparisons if not stated otherwise. This also means that correlations can differ from the results of the models presented by their respective authors as we want to only compare the different models in general and not find the specific one that performs best, as this would extend the scope of this project too far.

5.1 BERTScore

Without specification, BERTScore works with different models depending on the language specified, e.g. *RoBERTa large* for English texts. Using BERT as the baseline we compare it with RoBERTa, ALBERT, DistilBERT and DeBERTa (He et al., 2020), a further improved version of BERT and RoBERTa using an enhanced decoder and disentangled attention. We wanted to compare with DeBERTa because the model was listed as high performing in the implementation[10].

For BERTScore, the WMT19 Dataset was used, see also Section 3. Because of timing constraints, we focused on German to English translations. To get more expressive results we iterated them multiple times. As BERTScore is a supervised metric, we used the human scores to get a correlation between the results and human annotators. The results can be found in Table 1. As can be seen, the bigger RoBERTa model performs slightly better than the baseline BERT, which could also explain its default usage in e.g. English translations. Still, other metrics can compete with the results of RoBERTa, with DeBERTa even outscoring it, as was hinted at in the implementation of BERTScore[10]. ALBERT performs worst score-wise, but this follows its smaller model structure. This structure on the other hand gives ALBERT a big advantage when comparing execution times or memory usage. When looking at the time, only ALBERT and DistilBERT are faster than BERT, which makes sense recalling their structure. Distil-

[4] http://www.statmt.org/wmt17/
[5] http://www.statmt.org/wmt19/
[6] https://tac.nist.gov/
[7] TAC08: https://tac.nist.gov/data/past/2008/UpdateSumm08.html
[8] TAC09: https://tac.nist.gov/data/past/2009/Summ09.html
[9] https://duc.nist.gov/
[10] https://github.com/Tiiiger/bert_score/blob/master/bert_score/utils.py

Model	Pearson Corr.	% Time BERT	% Memory BERT
bert-base-uncased	0.322	100.0	100.0
roberta-base	**0.330**	100.2	254.5
albert-base-v2	0.311	**85.7**	**13.9**
distilbert-base-uncased	**0.323**	**79.1**	**99.9**
microsoft/deberta-base	**0.339**	106.8	254.3

Table 1: Results for BERTScore on the WMT19 dataset for stated variants, comparing Pearson correlation, time- and memory-efficiency, the time and memory usage are shown relative to BERT for better comparison; **bold** numbers are comparatively better than BERT.

BERT here performs slightly better than ALBERT. RoBERTa and DeBERTa take slightly longer than BERT, but this is not their main downside, which is the memory usage. As they are larger models compared to BERT with more parameters even in their base versions, they both use about 2.5 times the memory of BERT. Here, DistilBERT performs similarly to BERT, while ALBERT profits from its lightness and uses less than 15% of the memory BERT uses.

5.1.1 BERTScore with Importance Weighting

We present the results of the models in BERTScore with importance weighting in Table 2. As can be seen there, the results are very similar to BERTScore without importance weighting, except for the correlations being higher in general. Using importance weighting influences ALBERT as it becomes comparatively faster, but more expensive on memory compared to BERT, while both RoBERTa and DeBERTa take longer and use more memory. DistilBERT is barely affected at all in its comparison to BERT.

5.2 MoverScore

For the evaluation of MoverScore, we take the same approach as for BERTScore, comparing the newer and competitive models of RoBERTa, ALBERT, and DistilBERT with the usual BERT model. For clarification, MoverScore uses not BERT but DistilBERT as a default model, which will be taken into consideration in the conclusion.

MoverScore was evaluated on the machine translation part of the WMT19 dataset (see 3), because of constraints from the environment (only limited GPU-usage) only a subset of the german to english machine translations could be used. The results of the experiments are shown in Table 3.

As can easily be seen, the choice to use DistilBERT instead of BERT makes a lot of sense performance-wise, as the correlation between the scores and human evaluations are just slightly worse, but the execution time and memory usage are about half as high as for BERT-base. For ALBERT and RoBERTa we receive interesting results that we cannot explain. ALBERT on one hand takes more time and memory compared to BERT, which should both be the other way around, while also performing considerably worse. This could be because of the limited environment we worked in, which could have led to less robust results. Nevertheless, we used the same data and reference scores for all models. RoBERTa on the other hand unexpectedly uses less memory than BERT while being about 25% slower and performing better than ALBERT but still much worse than even normal BERT. Here, the timing could be explainable, but the memory usage and also correlations should be better by a margin.

For further work on these inconsistencies, a bigger setup with more data would be able to show if the tendencies remain. If so, using other models than the plain *base* models used could also yield more consistent results. The approximation to the word mover's distance (as proposed by Kusner et al. (2015)) used in MoverScore would also be able to indicate the reasons for our results, but for this, we have to refer to future work as we could not set it up in the given time.

5.3 XMoverScore

To evaluate XMoverScore for different transformer models and their performance, we spectate their corresponding score, time consumption, and memory usage for the plain score and the suggested remapping score established by the additional language model. In our case, we uniformly used "GPT2". The scores were evaluated on a pre-chosen selection of language pair for the Machine Translation task. The

Model	Pearson Corr.	% Time BERT	% Memory BERT
bert-base-uncased	0.349	100.0	100.0
roberta-base	**0.355**	111.5	276.7
albert-base-v2	0.344	**79.1**	**35.9**
distilbert-base-uncased	**0.350**	**79.7**	**99.0**
microsoft/deberta-base	**0.359**	115.5	275.7

Table 2: Results for BERTScore using importance weighting (see section 2.3 for details). Time and memory usage are shown relative to BERT for better comparison; **bold** numbers are comparatively better than BERT.

Model	Pearson Corr.	% Time BERT	% Memory BERT
bert-base-uncased	0.336	100.000	100.000
albert-base-v2	0.251	130.070	172.844
distilbert-base-uncased	0.318	**52.864**	**52.655**
roberta-base	0.285	123.206	**97.942**

Table 3: Results for MoverScore on the WMT19 dataset . Machine-dependent results are shown relative to BERT; **bold** numbers are comparatively better than BERT.

language pairs are `["cs-en", "de-en", "fi-en", "lv-en", "ru-en", "tr-en", "zh-en"]` . From here on abbreviated without the "en" suffix.

The results for the machine translation task into english are presented in a star plot in figure 4. The Pearson correlation value is calculated with the XMoverScore and a human score.

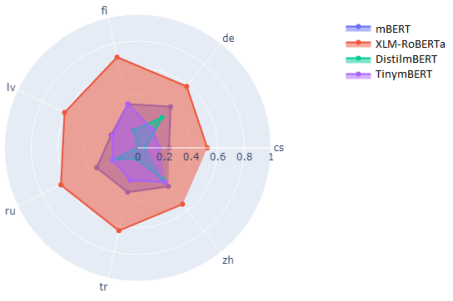

Figure 4: Pearson correlation for XMoverScore **without** additional LM on WMT17; the dots symbolise a value between 0 and 1 for a certain language paired with english; 0 being the middle and 1 being the outer shell

The first impression we get is that all transformers perform similarly in their range of performances regardless of the language. Only DistilmBERT seems to have performance drops in "lv"- , "tr"- and "cs"-"en". This observation brings near the assumption that these languages are not supported. However the model description suggests otherwise [11] . Further, we can see that XLM-RoBERTa-base[12] performs the best with quite a high lead in all language pairs. TinymBERT [13] achieves the same performance as its predecessor "mBERT" for "fi" , "lv" and "zh", while the other languages performs weaker.

In figure 5 we can see a different star plot with the performance scores of the transformers coupled with an additional language model. The scores for each language pair behave similarly for each language pair. The attentive reader will notice that adding Language Model (LM) causes a significant performance increase. Also, models, which did not behave uniformly for all language pairs, do so after adding the language model for remapping. XLM-RoBERTa-base still outperforms the other transformers.

In figure 6 the performance change from adding an additional LM is shown. We can see that all models have profited significantly from the addition LM with DistilmBERT the most. While XLM-

[11] https://huggingface.co/distilbert-base-multilingual-cased

[12] https://huggingface.co/xlm-roberta-base

[13] https://github.com/huawei-noah/Pretrained-Language-Model/tree/master/TinyBERT

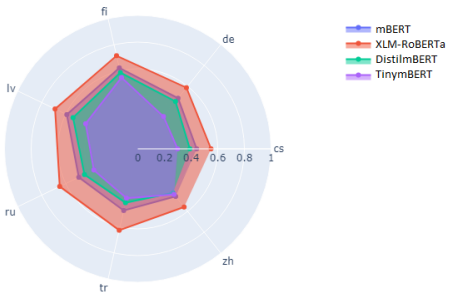

Figure 5: Pearson correlation for XMoverScore **with** additional LM on WMT17; the dots symbolise a value between 0 and 1 for a certain language paired with english; 0 being the middle and 1 being the outer shell

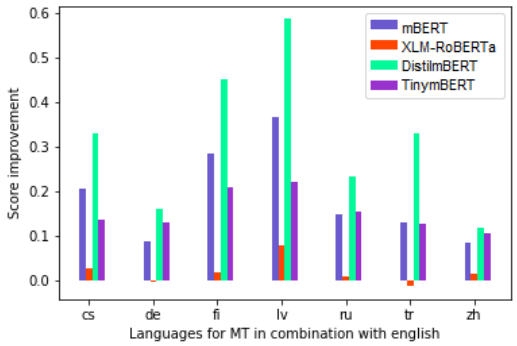

Figure 6: Improvement of Pearson Correlation when added a LM for remapping

RoBERTa-base seems quite unaffected.

The efficient multilingual BERT-variants perform better with a additional language model for remapping. DistilmBERT profits in particular of the additional language model while XLM-RoBERTa-base behaves unaffected.

In figure 7 the performance-related attributes in relation to mBERT are shown. The horizontal red line indicates the baseline for mBERT. All the used transformer for this task seem to work faster and consume the same or fewer memory storage. On the other hand the only model which outperforms mBERT score-wise is XLM-RoBERTa-base, while the others score lower.

The last two attributes of the plot show the measured time and memory consumption for only evaluating the additional language model. It is no surprise that these measure the same since the language model did not change for all experiments. In table 6 of Appendix A we provided the plain

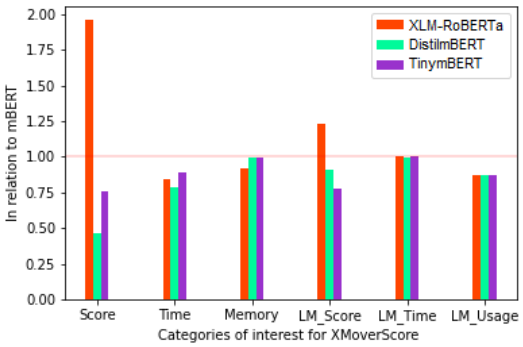

Figure 7: Results for general performance evaluated on XMoverScore

values of our evaluation. It can be noticed that the value gaps in the different language pairs are varying strong. Especially for TinymBERT some language pairs seem to take up to 50% of the the time and the memory in relation to mBERT, while other languages pairs like ”ru”’-”en” charge even more.

Our evaluation showed that using efficient variants of BERT can save up to 22% of time in average in calculating the XMOVERScore, but with significant losses in score. The only transformer which seems to perform fast, computationally efficient and better in average in score is XLM-RoBERTa-base.

5.4 SUPERT

To evaluate SUPERT it will be executed with the given BERT-Variations which are compatible with the given executable code for this metric. In this case, they are BERT as the baseline, RoBERTa, and DistilBERT. Experimenting with other BERT variations was not possible due to technical incompatibility with the available implementation of SUPERT. The used dataset is described in Section 3. Like in the other metrics the memory consumption and the execution time will also be tracked and compared. To compare its score SUPERT with the different BERT variations is used as a reward function for a reinforcement learning summarizer. The generated summaries are compared to given human reference summaries with ROUGE as a metric.

The experiments were executed 10 times and the average values of all runs of the experiments were taken. Table 4 shows the results of the experiments with SUPERT. Table 5 shows the ROUGE-Values of using SUPERT with diffferent BERT variations as a reward function. Executing the metric with

Model	% Time BERT	% Memory BERT
bert-base-nli-stsb-mean-tokens	100.0	100.0
roberta-base-nli-stsb-mean-tokens	102.4	207.4
distilbert-base-nli-stsb-mean-tokens	**52.0**	**97.2**

Table 4: Execution performance for SUPERT

Model	ROGUE-1	ROGUE-2	ROGUE-L	ROGUE-SU4
bert-base-nli-stsb-mean-tokens	0.285	**0.051**	**0.153**	0.071
roberta-base-nli-stsb-mean-tokens	0.298	0.036	0.132	0.071
distilbert-base-nli-stsb-mean-tokens	**0.315**	0.040	0.140	**0.084**

Table 5: ROGUE-Results for SUPERT

RoBERTa takes roughly the same amount of time as using BERT for this metric. The memory consumption is approximately doubled compared to using the metric with BERT. Meanwhile, executing SUPERT with DistilBERT reduces the execution time by roughly about a half compared to using BERT. The memory consumption is approximately the same compared to the memory usage when using the metric with BERT. Looking at the ROGUE-Scores DistilBERT scores similar to BERT while RoBERTa has in average the worst performance of all three tested BERT variations.

5.5 COMET

Unfortunately, the implementation of the COMET framework is not transparent enough to replace the BERT model with a more efficient version of it at a time that is suitable for the scope of this project. That is why we have to leave it out and cannot present results for it.

6 Discussion

We expected that with the use of more efficient BERT versions in the metrics we would get a lower score and faster and uses less memory. But in some cases the small versions can perform even better for the general score. Especially DistilBERT shows very good results. This could be one reason why MoverScore for example uses DistilBERT by default. We also see some unexpected results in small models like ALBERT that use more memory and takes longer than the baseline BERT model, while RoBERTa takes less memory than the baseline BERT model for MoverScore. We explain this with changing conditions for example that other processes run in parallel. Here it should be tested more often to get a more stable average of the results.
Nevertheless, most experiments show consistent results that were expected and sometimes exceeded. Depending on the specific use case we recommend considering different models to get more desired results.

7 Conclusion

All in all, we have shown that metrics do benefit from using lightweight BERT versions to run faster, use less memory and in some cases show better results. This implies that future metrics benefit from using the individually most fitting model for their goals. Furthermore, the addition of new Transformer-models will give more opportunities in the future, those models need to be monitored and considered as well for efficient metrics.
To get more confident results further experiments are needed and newer even faster BERT versions can be tested to improve the performance of the metrics even more.

Acknowledgements

Paul conducted the experiments on BertScore and MoverScore and wrote the abstract and sections 1, 2.2, 2.3, 2.4 5.1, 5.2, 6. Sewin conducted the experiments on XMoverScore and wrote sections 2.1, 2.2, 2.5, 3, 5.3. Leon Wempe conducted the experiments on COMET and wrote sections 2.6, 5.5, 6, 7. Dwarak conducted the experiments on SUPERT and wrote sections 2.1, 2.7, 3, 5.4.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings*

of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization, pages 65–72.

Arup Baruah, Kaushik Das, Ferdous Barbhuiya, and Kuntal Dey. 2020. Iiitg-adbu at semeval-2020 task 12: Comparison of bert and bilstm in detecting offensive language. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1562–1568.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of Bleu in machine translation research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy. Association for Computational Linguistics.

Miruna Clinciu, Arash Eshghi, and Helen Hastie. 2021. A study of automatic metrics for the evaluation of natural language explanations. *arXiv preprint arXiv:2103.08545*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *Nips, modern machine learning and natural language processing workshop*, volume 2.

Yang Gao, Wei Zhao, and Steffen Eger. 2020. SUPERT: towards new frontiers in unsupervised evaluation metrics for multi-document summarization. *CoRR*, abs/2005.03724.

Dimitra Gkatzia and Saad Mahamood. 2015. A snapshot of NLG evaluation practices 2005 - 2014. In *Proceedings of the 15th European Workshop on Natural Language Generation (ENLG)*, pages 57–60, Brighton, UK. Association for Computational Linguistics.

Yong Guo, Yin Zheng, Mingkui Tan, Qi Chen, Jian Chen, Peilin Zhao, and Junzhou Huang. 2019. Nat: Neural architecture transformer for accurate and compact architectures. *arXiv preprint arXiv:1910.14488*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling BERT for natural language understanding. *CoRR*, abs/1909.10351.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2021. Lightmbert: A simple yet effective method for multilingual bert distillation.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966. PMLR.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

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.

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

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic Turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1116–1126, Vancouver, Canada. Association for Computational Linguistics.

Nitika Mathur, Timothy Baldwin, and Trevor Cohn. 2019. Putting evaluation in context: Contextual embeddings improve machine translation evaluation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2799–2808, Florence, Italy. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Debora Nozza, Federico Bianchi, and Dirk Hovy. 2020. What the [mask]? making sense of language-specific bert models. *arXiv preprint arXiv:2003.02912*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. Comet: A neural framework for mt evaluation. *arXiv preprint arXiv:2009.09025*.

Ehud Reiter and Anja Belz. 2009. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4):529–558.

Ananya B. Sai, Akash Kumar Mohankumar, and Mitesh M. Khapra. 2020. A survey of evaluation metrics used for nlg systems.

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

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. Bleurt: Learning robust metrics for text generation. *arXiv preprint arXiv:2004.04696*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2019. BERTScore: Evaluating Text Generation with BERT. *arXiv e-prints*, page arXiv:1904.09675.

Ying Zhang, Stephan Vogel, and Alex Waibel. 2004. Interpreting bleu/nist scores: How much improvement do we need to have a better system? In *LREC*.

Wei Zhao, Goran Glavaš, Maxime Peyrard, Yang Gao, Robert West, and Steffen Eger. 2020a. On the limitations of cross-lingual encoders as exposed by reference-free machine translation evaluation. *arXiv preprint arXiv:2005.01196*.

Wei Zhao, Goran Glavaš, Maxime Peyrard, Yang Gao, Robert West, and Steffen Eger. 2020b. 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. *arXiv preprint arXiv:1909.02622*.

A Appendices

	LP	Score	LM-Score	% Time of BERT	% Memory of BERT	% LM_Time of BERT	% LM_Memory of BERT	% Avg. Time of BERT	% Avg. Memory of BERT
mBERT	cs-en	0.235	0.442	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%
	de-en	0.395	0.485	100,0%	100,0%	100,0%	100,0%		
	fi-en	0.337	0.622	100,0%	100,0%	100,0%	100,0%		
	lv-en	0.222	0.591	100,0%	100,0%	100,0%	100,0%		
	ru-en	0.342	0.491	100,0%	100,0%	100,0%	100,0%		
	tr-en	0.341	0.474	100,0%	100,0%	100,0%	100,0%		
	zh-en	0.369	0.456	100,0%	100,0%	100,0%	100,0%		
mDistilBERT	cs-en	0.058	0.391	74,6%	97,7%	99,5%	81,0%	77,2%	99,4%
	de-en	0.291	0.454	77,5%	99,0%	100,1%	81,0%		
	fi-en	0.134	0.586	73,8%	99,5%	100,1%	67,8%		
	lv-en	-0.049	0.539	81,3%	99,8%	99,2%	82,0%		
	ru-en	0.209	0.444	79,3%	99,9%	**98,6%**	100,0%		
	tr-en	0.081	0.414	77,4%	100,0%	100,5%	100,1%		
	zh-en	0.302	0.422	76,3%	99,9%	100,5%	100,0%		
XLM-RoBERTa-base	cs-en	0.522	0.550	84,7%	87,7%	99,6%	81,0%	84,1%	97,2%
	de-en	0.589	0.586	97,6%	114,4%	100,6%	81,0%		
	fi-en	**0.697**	**0.716**	83,2%	129,9%	101,0%	67,8%		
	lv-en	0.610	0.689	80,4%	80,8%	99,3%	82,0%		
	ru-en	0.640	0.650	86,8%	94,7%	100,3%	100,0%		
	tr-en	0.637	0.626	80,3%	79,0%	102,2%	100,0%		
	zh-en	0.540	0.558	75,8%	93,7%	101,0%	100,0%		
TinyBERT	cs-en	0.161	0.299	61,0%	**54,3%**	100,5%	81,0%	88,8%	99,7%
	de-en	0.178	0.310	77,6%	85,9%	99,9%	81,0%		
	fi-en	0.338	0.548	70,1%	77,9%	101,3%	**67,7%**		
	lv-en	0.210	0.434	**52,5%**	69,0%	99,7%	82,0%		
	ru-en	0.210	0.366	208,3%	240,1%	98,7%	100,0%		
	tr-en	0.250	0.379	74,7%	86,5%	100,8%	100,0%		
	zh-en	0.330	0.437	77,5%	84,2%	100,6%	100,0%		

Table 6: XMoverScore results; Values for Time and Memory Consumption are stated in relation to BERT for comparability; best values are marked **bold**