A Fine-Grained Analysis of BERTScore

Michael Hanna
Charles University
hannamichaelw@gmail.com

Ondřej Bojar
Charles University
ondrej.bojar@mff.cuni.cz

Abstract

BERTScore (Zhang et al., 2020), a recently proposed automatic metric for machine translation quality, uses BERT (Devlin et al., 2019), a large pre-trained language model to evaluate candidate translations with respect to a gold translation. Taking advantage of BERT's semantic and syntactic abilities, BERTScore seeks to avoid the flaws of earlier approaches like BLEU, instead scoring candidate translations based on their semantic similarity to the gold sentence. However, BERT is not infallible; while its performance on NLP tasks set a new state of the art in general, studies of specific syntactic and semantic phenomena have shown where BERT's performance deviates from that of humans more generally.

This naturally raises the questions we address in this paper: what are the strengths and weaknesses of BERTScore? Do they relate to known weaknesses on the part of BERT? We find that while BERTScore can detect when a translation exhibits poor choice of content words, it struggles to correctly penalize incorrect candidates when they are lexically similar to the reference, or only differ in function words.

1 Introduction

While manual, human evaluation of machine translation (MT) systems is still the gold standard, automatic evaluation metrics have long been used for their relative speed and inexpensiveness. Early automatic metrics were easy to implement and correlated with human judgements, but have clear limitations: BLEU (Papineni et al., 2002) relies on n-gram overlap, and is thus not robust to differing word order or choice. In contrast, METEOR (Lavie and Agarwal, 2007) requires training, but depends on token alignment, which is also a fraught task.

With the advent of deep learning, new automatic metrics have arisen, both in response to and making use of the technical advances brought by deep learning. In particular, metrics like COMET (Rei et al., 2020) and BERTScore use large pre-trained language models (LLMs) to generate scores for candidate sentences. The use of these LLMs allows for metrics that take advantage of the linguistic capabilities of these LLMs, and no longer rely solely on surface-level features such as n-grams.

The expressiveness of these models is both a boon and a danger. While they can (and do, based on correlation with human judgments) generate more useful scores for translations, how they arrive at the score, and which types of sentences they will score accurately is not immediately obvious.

Moreover, these LLMs are known to have flaws. BERT in particular has been shown to be, in certain scenarios, insensitive to negation (Ettinger, 2020) and word order (Pham et al., 2020). BERT also has inexact representations of numbers (Wallace et al., 2019) and fails to be robust to named entities (Balasubramanian et al., 2020). All of these phenomena could result in poor-quality scores from BERTScore. However, it is difficult to say for certain how these issues might manifest in BERTScore, as it employs BERT in an unsupervised scenario distinct from that of these analyses.

Thus, in this paper, we analyze BERTScore. We first formally define desiderata for a MT metric. Then, we consider how BERTScore fulfills these requirements under conditions of interest. We find that BERTScore violates some of these requirements, specifically the requirement that incorrect translations be rated below correct ones; this occurs most often when the incorrect translation is lexically similar to the reference, or especially if the difference is only in function (not content) words.

2 Desiderata for MT metric quality

The most common method of measuring the quality of a MT metric is correlation with human judgments (Fomicheva and Specia, 2019); however, these correlations provide little information regarding when and why an MT metric differs from human judgment. In this paper, we consider three ways of examining MT metric quality, with the aim of determining the failure cases of MT metrics.

In all of our experiments, we assume the following setup. We have a MT metric, which we take to be a function M that takes as input a reference and candidate translation, and outputs a score in $[0, 1]$. We also have a dataset $\mathcal{D}$, consisting of triples (x, Y, B) where x is a source sentence, Y is a list of at least two reference translations, and B is a list of at least one "bad" translation, which contains errors.

Then we state that a good MT metric[1] M fulfills, for any triple $(x, Y, B) \in \mathcal{D}$, where $Y = y_1, y_2, \ldots, y_n$, and $B = b_1, b_2, \ldots, b_m$, the following conditions:

(i) For any pair (y_i, y_j) of reference translations, $M(y_i, y_j) \approx M(y_j, y_i) \approx 1$.

(ii) For any reference translation y_i and candidate translation b_i, $M(y_i, b_i) < 1$ and $M(b_i, y_i) < 1$. It follows that given another reference translation y_j, we should have that $M(y_i, b_i) < M(y_i, y_j)$ and so on.

(iii) If we know the relative quality of the bad translations in B, let B be sorted in increasing order of translation quality. Then if y_i is any reference translation, and b_i, b_j are bad translations in B where $i < j$, $M(y_i, b_i) < M(y_i, b_j)$.

Put simply, reference translations should be scored near 1 when compared to each other (i), bad candidate translations should be scored worse than reference translations (ii), and the scoring of bad candidate translations should reflect their relative quality (iii).

To use this framework to investigate the failure points of MT metrics, we simply need a dataset that contains phenomena of interest; for example, we might be interested in knowing if a MT metric is able to distinguish between translations that do and do not correctly render negation. Then, we simply compute the quantities discussed in conditions (i) through (iii) for each example, and see which, if any, conditions are violated. If, for example, condition (i) is violated when two equivalent references employ different types of negation, this might imply that our metric is not robust to this sort of negation phenomenon.

[1]We assume WLOG that the metric's scores are normalized such that better translations receive higher mean scores, and equally good candidates receive a score near 1.

Note that throughout these desiderata only concern the scores given to the reference and candidate sentences; the source-language sentence is ignored. This is because, since the MT metric operates only on the target language, it is not strictly necessary that the reference / bad candidate translations be translations of any source sentence at all. It is possible to construct a dataset of only reference and bad candidate translations that exhibit phenomena of interest. However, it may be desirable to use real translations, so that the dataset reflects the distribution of real-world translation errors.

3 BERTScore and BERT

3.1 BERTScore

In this paper, our metric of interest is BERTScore (Zhang et al., 2020). To compute BERTScore, we first feed a reference and candidate translation for a given sentence into BERT, and retrieve their token level vector representations. Let x be the representations of the reference and $\hat{x}$ those of the candidate. Then we compute the precision and recall metrics for BERTScore by comparing each token representation x_i of the reference translation to each token representation $\hat{x}_j$ of the candidate translation as follows:

$$P_{BERT} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} x_i^\top \hat{x}_j$$

$$R_{BERT} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} x_i^\top \hat{x}_j$$

The F_1 score can be defined as usual. As BERTScore can range from -1 to 1, but most often inhabits the upper end of that range, its creators suggest the use of baseline scaling, which generally leaves BERTScore in the range [0,1], as desired for use with our prior formalization. Baseline rescaling is performed for P_{BERT} as

$$\hat{P}_{BERT} = \frac{P_{BERT} - b}{1 - b}$$

and likewise for R_{BERT}; b is an empirical lower bound on observed BERTScore.

The form of BERTScore naturally leads to its interpretation as a similarity-based metric. It penalizes candidates containing words whose representations are not similar to any of the reference's words' representations (precision), and vice-versa (recall). As a result, the quality and characteristics of these representations, derived from BERT, will play a key role in the quality of BERTScore.

3.2 BERT

What, then, is known about BERT, and its syntactic and semantic capabilities? Of the two, it is syntax that BERT is most widely claimed to capture within its internal representations: Hewitt and Manning (2019) use structural probing to find dependency trees in BERT's vector geometry, while Tenney et al. (2019) use probing to find part of speech tags and dependency arc labels, among other types of syntactic information. Analysis of BERT's attention has shown that certain heads attend to not only relevant linguistic units such as determiners of nouns and coreferent mentions (Clark et al., 2019), but also dependency relations (Htut et al., 2019).

However, these analyses of internal representations and the information contained therein occasionally come at odds with targeted evaluations of BERT's syntactic abilities. Despite BERT's supposed knowledge of syntax, its predictions often remain the same, even when its inputs are shuffled (Pham et al., 2020). Moreover, BERT does not seem to understand negation (Ettinger, 2020); this may be due to BERT encoding syntactic information, but not necessarily using it in its predictions (Glavas and Vulić, 2021).

For semantics, the situation is even more complicated. While BERT's performance on natural language understanding tasks set a new state of the art, more targeted tests of its semantic abilities have yielded less positive results. BERT has limited knowledge of lexical semantic relations such as hypernymy (Ravichander et al., 2020) and antonymy (Staliunaite and Iacobacci, 2020). Moreover, it has fragile representations of named entities (Balasubramanian et al., 2020), and imprecise representations of numbers (Wallace et al., 2019). In sum, these flaws provide a starting point, specific linguistic phenomena that BERTScore, due to its use of BERT, maybe be unable to handle.

4 Experiments

For our experiments, we utilize the framework described in Section 2 to investigate the questions about BERT described in Section 3. As the number of datasets that fit our framework is few, we limit ourselves to three primary experiments.

German Source: Ich habe mich konzentriert.
Good Translation 1: I focused.
Good Translation 2: I've been concentrating.
Bad Translation 1: I've focused me.
Bad Translation 2: I have focussed.
Broad phenomenon: Verb Tense / Aspect / Mood
Specific phenomenon: Reflexive - Perfect

Figure 1: Example data from TQ-AutoTest.

4.1 TQ-AutoTest

First, we apply our framework to the TQ-AutoTest dataset (Macketanz et al., 2018). Originally used for targeted evaluation of MT systems, it includes German source sentences that each exhibit one of 14 different linguistic phenomena, such as ambiguity, composition, and subordination.

Each example contains a source sentence, annotated with the broader and more specific phenomenon it exhibits, as well as 1-2 reference translations and 0-2 incorrect translations; see Figure 1 for a sample. As the dataset released is small, we filter out phenomena containing fewer than 5 examples.

We can thus test condition (i) by computing the BERTScore between the two references, and verifying it is close to 1. We also test condition (ii) by comparing the BERTScore assigned (with respect to a given reference) to the other reference, and that which is assigned to a bad translation; the former should be greater than the latter.

4.2 PE^{2}rr

Second, we use the PE^{2}rr dataset (Popović and Arčan, 2016), which is a manually annotated error analysis of MT output. Each example in the dataset consists of a source sentence, one MT output, and two correct translations, along with two error annotations. These annotations are word-level annotations into 8 broadly non-linguistic classes, such as "addition", "lexical error", or "untranslated". See Figure 2 for an example.

Despite the difference in annotation type, we apply our framework just as with the prior experiment. For consistency with the prior experiment, we use only the portion of the dataset where the target language is English (i.e., the German-English portion). We also filter out any examples in which the machine translation output is totally correct,

German Source: Frauen , die in Burkina Faso zu Hexen abgestempelt werden , weisen in der Regel einige gemeinsame gesellschaftliche Merkmale auf .
Original Translation (Annotated): Women in Burkina Faso `[miss]` are branded as witches , usually `[miss]` some common social features .
Post-edit 1: Women in Burkina Faso who are branded as witches usually have some common social features .
Post-edit 2: Women declared as witches in Burkina Faso usually have several common characteristics .

Figure 2: Example data from PE^{2}rr. The "`[miss]`" tokens inserted in the original translation correspond to one of the annotator's error notations, indicating a missing word.

as this would leave no bad examples with which to test conditions other than (i). This once more allows us to test conditions (i) and (ii).

4.3 Grammatical Error Correction

Finally, we use two non-MT datasets for grammatical error correction (GEC): the CoNLL 2014 shared task dataset (Ng et al., 2014), as well as additional annotations released by Bryant and Ng (2015). The former consists of non-native English speakers' essays on genetic testing, paired with annotated corrections from two annotators for each sentence. The latter adds additional annotators' corrections for each sentence, yielding 11 in total.

Of special interest in this dataset is the presence of, for each example, an incorrect sentence and 11 sets of error-annotated corrections that can be applied to obtain correct sentences. Thus, for each original, ungrammatical sentence, we have 10 (often distinct) grammatical sentences, whose meaning should be roughly the same; these act as reference sentences. This allows us to test conditions (i) and (ii).

We can also test condition (iii) by using the following moderate assumption: a sentence, originally grammatically incorrect, is more correct if more corrections have been applied to it. That is, applying one or more corrections (from the same annotator) brings an incorrect candidate sentence closer to the shared meaning of the reference sentences. This assumption can be false: sometimes a group

Original: As a result , if the situation keep go on in this unexpected trend , it will cause a bad effect on the young generation .
Corrections (Annotator 2):
(7, 8, 'keeps', subject-verb agreement)
(8, 10, 'following', word choice)
(10, 11, '', preposition)
(17, 18, 'have', word choice)
(23, 24, 'younger', word form)
Original + 1 correction (b_1): As a result , if the situation keeps go on in this unexpected trend , it will cause a bad effect on the young generation .
Original + 4 corrections (b_4): As a result , if the situation keeps following this unexpected trend , it will have a bad effect on the young generation .
Alternate reference (Annotator 7) (y): As a result , if the situation keeps going on in this unexpected trend , it will have a bad effect on younger generations .

Figure 3: Creation of graded ungrammatical sentences from GEC data. Each correction is a tuple of (start index, end index, new text, error made). We apply a correction via string replacement (i.e. in Python: `original[start_index: end_index] = new_text`). Having received more corrections, b_4 should be closer to y than b_1 is. Thus, we should have BERTScore(y, b_4) $>$ BERTScore(y, b_1).

of corrections, rather than one alone, is needed to increase the grammaticality of a sentence. But, this assumption allows us to apply an arbitrary number of corrections to an initially incorrect sentence, to generate intermediate incorrect sentences, with controlled, graded, levels of incorrectness.

So, we generate two incorrect candidates by applying different numbers of edits from the same annotator to one original sentence. We then generate a reference sentence by applying all of another annotator's edits. Finally, we calculate the BERTScore of each candidate with respect to this reference; the candidate that received more edits should receive a higher BERTScore. Note that the reference must be generated by a different set of edits; comparing partially-corrected sentences to a correct sentence generated from same edits would make this a trivial string comparison problem. Figure 3 provides an example of this process.

Linguistic Phenomenon	BERTScore (F_1)
LDD & interrogatives	84.91
Composition	85.15
Punctuation	86.37
Function word	71.25
Subordination	76.26
Non-verbal agreement	83.02
Ambiguity	80.40
Verb tense/aspect/mood	79.50
Coordination & ellipsis	88.48
Named entitiy & term.	85.87
MWE	87.44
Average	81.55

Table 1: TQ-AutoTest: Mean BERTScore (F_1) assigned to gold reference, gold candidate pairs, by linguistic phenomenon

Linguistic Phenomenon	Accuracy
LDD & interrogatives	82.36
Composition	60.00
Punctuation	40.00
Function word	42.86
Subordination	75.00
Non-verbal agreement	60.00
Ambiguity	100.00
Verb tense/aspect/mood	73.61
Coordination & ellipsis	80.00
Named entity & terminology	85.71
MWE	100.00
Average	75.46

Table 2: TQ-AutoTest: Average Accuracy by broad linguistic phenomenon. Credit is given when BERTScore(y_1, y_2) is greater than BERTScore(y_1, b_1), and BERTScore(y_1, b_2), if a b_2 is provided.

5 Results

In the following section, we detail the experiments performed and their results. Note that in all experiments, the original authors' implementation of BERTScore[2] is used, with default baseline rescaling.

5.1 TQ-Autotest

As discussed in Section 4.1, we test conditions (i) and (ii) with the TQ-Autotest dataset. First, we filter the dataset to include only those examples for which there are at least two good translations (y, y'). Then, to test condition (i), we compute the BERTScore (BERTScore(y, y')) assigned to the pair of good translations, then compute the mean score for each category. Note that as we use baseline rescaling, we should not make absolute comparisons between the BERTScores assigned and the desired value (1.0) of the scores. Rather, it is useful to see in which categories of linguistic phenomenon BERTScore is more or less able to assign a high score despite differences in the surface form of the translation.

We see that the BERTScore assigned per category falls near the average, 80. There are some notable exceptions: the "function word" category, which falls well below the mean, indicating that BERTScore gives different translations of the same sentence lower scores. In contrast, the "coordination & ellipses" and "multi-word error" categories fall well above the mean.

In the second experiment, we test condition (ii) via the following procedure. Once more, we filter the dataset, such that every example includes two good translations and one bad translation, (y, y', x), with potentially another bad translation x' as well. Then, we test the accuracy of BERTScore on these examples. BERTScore is deemed to correctly answer an example if BERTScore$(y, y') >$ BERTScore(y, x). If x' is present, as in 71% of examples, it is also necessary that BERTScore$(y, y') >$ BERTScore(y, x'). We then report the mean accuracy for each category of linguistic phenomenon.

In this second experiment the differences are more pronounced. In some categories, namely those such as "ambiguity", and "multi-word error" which are likely to result in totally incorrect word choice (i.e. an obvious lexical difference), BERTScore has a high accuracy. In contrast, BERTScore struggles with difficult punctuation as well as composition errors and function words; notably, the last category also the lowest score in Table 1. These errors are all somewhat subtle. Errors in function words are by definition not errors in more obvious content words. Similarly, in compositional phenomena like phrasal verbs, an error can appear simply as the omission or incorrect substitution of a mere preposition.

5.2 PE^{2}rr

Using the PE^{2}rr dataset, we again test conditions (i) and (ii). We use an approach like that taken with the TQ-AutoTest dataset. First, we filter out examples in which the two references translations provided

[2]Available at https://github.com/Tiiiger/bert_score

Error Type	Count	BERTScore (F_1)
reordering	135	0.56
untranslated	55	0.55
lexical	171	0.58
inflection	72	0.57
derivation	16	0.54
missing	164	0.58
contraction	14	0.56
Average	90	0.59

Table 3: PE^{2}rr: Mean BERTScore (F_1) for sentences containing at least one error of the given type, and the counts of these sentences.

Error Type	Acc. (Easy)	Acc. (Hard)
reordering	0.97	0.44
untranslated	1.00	0.73
lexical	0.98	0.46
inflection	1.00	0.50
derivation	1.00	0.50
missing	0.97	0.46
contraction	1.00	0.50

Table 4: PE^{2}rr: Average Accuracy by error type, for both the easy and hard problem scenarios. Credit is given when BERTScore(y, y) is greater than BERTScore(y_1, b_1).

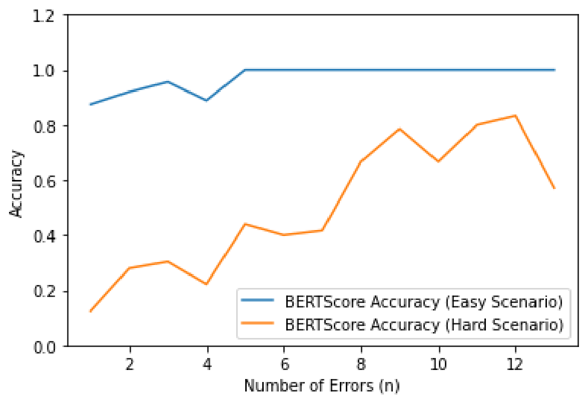

Figure 4: PE^{2}rr: BERTScore accuracy on sentences with n errors, in the easy and hard scenarios.

are identical; in this case, the BERTScore will be trivially 1. Then, for all other examples (y, y') we compute BERTScore(y, y'). Then, for each error category, we compute the mean BERTScore (F_1) among all examples that contain at least one of that error. Results are reported in Table 3.

We find that the average BERTScore assigned to pairs of correct translations in this dataset, 59.00, is much lower than in TQ-AutoTest, where the average BERTScore assigned to correct translations pairs was 81.55. Moreover, PE^{2}rr examples without any errors in their machine translation have a higher BERTScore assigned to their correct translations.

Testing condition (ii) with the PE^{2}rr dataset is somewhat more challenging. First, we filter out any of the machine translations that have no errors (as we need a bad translation to test condition (ii)). Normally, the next step would be to compute and compare, using our two references translations y, y' and one bad translation b, BERTScore(y, y') and BERTScore(y, b).

However, a difficulty arises: while each example has two correct translations, and one incorrect machine translation, the two correct translations are not generated in the same way. One is a post-edit of the machine translation, while the other is an original reference, generated from the German text without the machine translation.

If we choose the original reference to be y, and the post-edit to be y', this is a fair comparison; however, if we choose the post-edit to be y, the task becomes much more challenging. This is because BERTScore(y, b) will be comparing a post-edited machine translation to the original machine translation, and there will naturally be a good deal of overlap, even though b contains errors. So, we report results in two cases in Table 4: the easy case, where y is the original reference, and the hard case, where y is the post-edit.

This choice has a major effect on the ability of BERTScore to distinguish good translations from bad ones. When we choose the original reference as y, BERTScore has a high accuracy in all categories. In contrast, in the hard problem setting, BERTScore does poorly (at or below chance) in all categories except sentences with an untranslated word, which is likely easy for BERTScore to detect.

Finally, to provide an alternate explanation for the trend in BERTScore accuracies, we plot in Figure 4 the BERTScore accuracy for examples with n

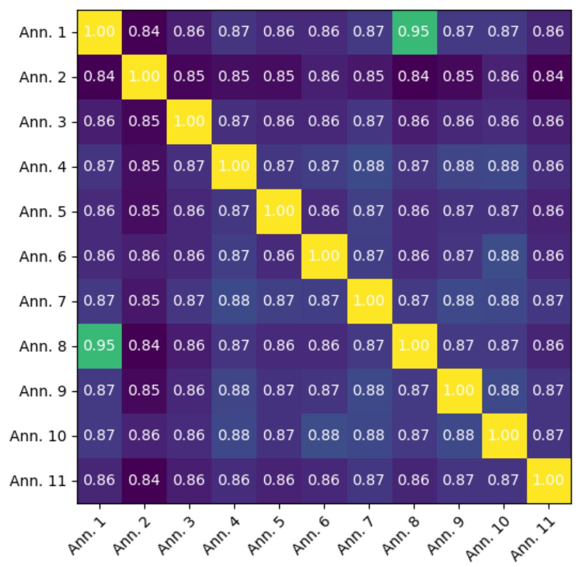

Figure 5: GEC: Heatmap of Average BERTScore (F_1) assigned when comparing references from a given pair of annotators

n: # of Errors	b_n: With Errors	b': Without Errors	y': Alternate Reference
1	0.841	0.850	0.865
2	0.830	0.844	0.860
3	0.820	0.839	0.856
4	0.814	0.828	0.847
5	0.812	0.821	0.842

Table 5: GEC: Average BERTScore (F_1) assigned when comparing a reference y to a) b_n, a sentence with all but n errors corrected, b) b', the same sentence, but with all errors corrected, and c) y', an alternate reference sentence

errors, in both the easy and hard scenarios. While in the easy scenario, BERTScore has an accuracy near 1 for $n >= 5$, in the hard scenario, the accuracy is lower, but increases with n. This suggests that it is easier for BERTScore to distinguish between translations that good and those that are bad, but lexically similar to the postedited reference, when the latter contain more errors.

5.3 Grammatical Error Correction

For GEC, we test all three conditions. To test the first, we use the 11 annotators' corrections to create post-edited versions $(y_1, \ldots, y_{11})$ of the original sentences; these should all have the same meaning. Then, we compute the mean BERTScore (F_1) for each pair of annotators' post edits; these are reported in Figure 5. The mean BERTScore tends to fall in a very narrow range (0.84-0.88), except for Annotators 1 and 8, who provided similar annotations.

We test conditions (ii) and (iii) jointly. To do so, we need two correct translations, created independently of one another. We also need two partially correct translations (of different levels of correctness) and one correct translation; these must be created from the same set of edits, independent of the first two. To create these, we first select three annotators, and assign them roles: reference annotator, alternate reference annotator, and "error" annotator. Next, for $n = 1, \ldots, 5$, we filter out all examples where the error annotator made fewer than n error annotations. Then, for each example, the reference and alternate reference annotators' annotations are used to create one reference sentence (y) and alternate reference sentence (y'). The error annotator's annotations are used to create both a third, fully corrected sentence (b'), and a sentence with n errors in it (b_n); the errors that remain are chosen at random.

Finally, we compute BERTScore(y, y'), BERTScore(y, b'), and BERTScore(y, b_n); the mean values for each of these is reported in Table 5. Note that although b' and y' have no errors, their mean BERTScore still changes with n because we filter out examples that have fewer than n errors according to the error annotator. Thus, the downward trend as n reflects the fact that, as the number of errors in a sentence increases, the annotators' corrections diverge.

We can see that BERTScore does respect conditions (ii) and (iii) within the GEC data. The alternate reference and error-free sentence are always assigned a higher score than the sentence with errors (ii). Moreover, the sentences with more errors are assigned a lower score than those with fewer (iii).

6 Discussion

From our experiments on three datasets, we draw three main findings. First, we find that the performance of BERTScore with respect to conditions (i) and (ii) can vary based on linguistic phenomena. Second, while BERTScore is generally capable, it has difficulties on a challenge dataset that tasks it with penalizing incorrect but lexically similar translations, and preferring lexically different but more correct translations. Third, BERTScore does, in certain circumstances, respect our third condition—it ranks worse bad translations below better bad translations.

With respect to the first finding, penalizing translations that incorrectly render function words seems to be the most difficult for BERTScore. In TQ-Autotest, this includes sentences with tag questions; in one example, the reference is “You’re crazy, aren’t you?”, and secondary good translation is “You’re crazy, right?”, while incorrect sentences are “You’re crazy, or?” and “You’re crazy, are not you?”. That is, the differences do not affect the main content of the sentence. In contrast, sentences with incorrectly resolved word ambiguity or larger, multi-word errors were easily penalized.

The second finding also confirms that BERTScore can more easily detect bad translations when there is less lexical overlap. In the easy problem setting, where both the good and bad candidate translations were lexically distinct from the reference, BERTScore easily distinguished the good translation from the bad. But, in the hard problem setting, where the bad translation had high lexical overlap with the reference, BERTScore struggled. Thus, BERTScore may be better-suited for scoring candidates from widely-differing systems, as opposed to closely-related systems, or multiple candidates from one system.

Our third finding, provides more positive results. In the GEC scenario, BERTScore was able to fulfill all three of the conditions we defined. It not only gave high scores to similar sentences and worse scores to sentences with errors; it also gave better scores to more grammatically correct sentences, even when they were not perfectly correct.

Unfortunately, the GEC dataset does not necessarily reflect the kinds of errors that occur in machine translation. It focuses primarily on grammatical errors, and thus contains fewer semantic errors. Moreover, since the GEC annotators had only the incorrect text, and no source text to work with, their annotations are occasionally in disagreement, as they each independently inferred the intended meaning of the incorrect text.

7 Related Work

The original paper introducing BERTScore (Zhang et al., 2020) naturally compared BERTScore’s correlations with human judgments to that of other metrics. However, various other surveys of MT metrics, as well as datasets and methodologies have been conducted, offering insights into how MT system and metric performance should be measured.

Naturally, the WMT Metrics task, most recently run in 2020 (Mathur et al., 2020b) is one such forum for the evaluation of metrics. In this last iteration, metrics were evaluated based on their correlation with human judgment scores on the sentence, paragraph, and document level. BERTScore was not included even in the most recent iteration of the metrics task.

More recently, Kocmi et al. (2021) run a large-scale comparison of MT metrics, including BERTScore using a large dataset of translations with human judgments; they find that BERTScore’s performance is middle-of-the-road, though better than BLEU, and recommend COMET (Rei et al., 2020) for general use.

Unfortunately, while these studies evaluate MT metrics, using human judgments alone cannot tell us when or why they may succeed or fail. In response to the results of 2019 WMT Metrics Task (Ma et al., 2019), Mathur et al. (2020a) note that correlations of metrics with human judgment can be highly sensitive to the number of systems in question, as well as outliers.

Fomicheva and Specia (2019) propose moving beyond correlation with human judgment alone as a standard for MT metric evaluation. To this end, they conduct a comprehensive study of MT metrics, and review datasets that have more fine-grained error and quality annotations. Despite this, datasets for MT metric evaluation with linguistic annotations or other annotations regarding sentence content are few.

8 Conclusion

BERTScore is a new metric for MT evaluation that uses BERT, and is as a result difficult to interpret. We define desiderata for BERTScore and other such metrics’ performance, and use targeted datasets to find when BERTScore fails. We find that BERTScore fails to assign low scores when a bad candidate sentence has high lexical overlap with the reference in terms of content words. Despite this, in less challenging scenarios, BERTscore does well, and is able to rank sentences in order of their quality. However, these experiments are limited in scope, due to limited available data with appropriate annotations. Development of datasets for MT metric evaluation with linguistic annotation would aid in further work on this topic.

References

Sriram Balasubramanian, Naman Jain, Gaurav Jindal, Abhijeet Awasthi, and Sunita Sarawagi. 2020. What's in a name? are BERT named entity representations just as good for any other name? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 205–214, Online. Association for Computational Linguistics.

Christopher Bryant and Hwee Tou Ng. 2015. How far are we from fully automatic high quality grammatical error correction? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 697–707, Beijing, China. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

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.

Allyson Ettinger. 2020. What BERT Is Not: Lessons from a New Suite of Psycholinguistic Diagnostics for Language Models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

Marina Fomicheva and Lucia Specia. 2019. Taking MT Evaluation Metrics to Extremes: Beyond Correlation with Human Judgments. *Computational Linguistics*, 45(3):515–558.

Goran Glavas and Ivan Vulić. 2021. Is supervised syntactic parsing beneficial for language understanding tasks? an empirical investigation. In *EACL*.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. 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 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R. Bowman. 2019. Do attention heads in bert track syntactic dependencies? *ArXiv*, abs/1911.12246.

Tom Kocmi, Christian Federmann, Roman Grundkiewicz, Marcin Junczys-Dowmunt, Hitokazu Matsushita, and Arul Menezes. 2021. To ship or not to ship: An extensive evaluation of automatic metrics for machine translation.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, page 228–231, USA. Association for Computational Linguistics.

Qingsong Ma, Johnny Wei, Ondřej Bojar, and Yvette Graham. 2019. Results of the WMT19 metrics shared task: Segment-level and strong MT systems pose big challenges. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 62–90, Florence, Italy. Association for Computational Linguistics.

Vivien Macketanz, Renlong Ai, Aljoscha Burchardt, and Hans Uszkoreit. 2018. TQ-AutoTest – an automated test suite for (machine) translation quality. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Nitika Mathur, Timothy Baldwin, and Trevor Cohn. 2020a. Tangled up in BLEU: Reevaluating the evaluation of automatic machine translation evaluation metrics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4984–4997, Online. Association for Computational Linguistics.

Nitika Mathur, Johnny Wei, Markus Freitag, Qingsong Ma, and Ondřej Bojar. 2020b. Results of the WMT20 metrics shared task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 688–725, Online. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. 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 *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Thang M. Pham, Trung Bui, Long Mai, and Anh Nguyen. 2020. Out of order: How important is the sequential order of words in a sentence in natural language understanding tasks?

Maja Popović and Mihael Arčan. 2016. PE2rr corpus: Manual error annotation of automatically pre-annotated MT post-edits. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 27–32, Portorož, Slovenia. European Language Resources Association (ELRA).

Abhilasha Ravichander, Eduard Hovy, Kaheer Suleman, Adam Trischler, and Jackie Chi Kit Cheung. 2020. On the systematicity of probing contextualized word representations: The case of hypernymy in BERT. In *Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics*, pages 88–102, Barcelona, Spain (Online). Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

Ieva Staliunaite and Ignacio Iacobacci. 2020. Compositional and lexical semantics in roberta, bert and distilbert: A case study on coqa. In *EMNLP*.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. 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 5307–5315, Hong Kong, China. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert.