LiSAScore: Exploring Linear Sum Assignment on BertScore

Stephen Mander[0009−0005−8073−0410]* and Jesse Phillips[0009−0001−8031−4864]

School of Computing and Communications, Lancaster University, UK
{s.mander3*, j.m.phillips}@lancaster.ac.uk

Abstract. Metrics play a crucial role in evaluating the performance of machine learning models. In the context of Natural Language Processing (NLP) tasks, such as text summarization and machine translation, Natural Language Generation (NLG) metrics such as Bleu and Rouge have been widely used. However, these metrics are based on n-gram matching and do not capture the semantic similarity between the generated and reference texts. To address this, BertScore has emerged as a popular evaluation metric that uses a pre-trained Large Language Model (LLM) to measure semantic similarity between two sentences. Unlike n-gram-based metrics, BertScore uses the contextual and semantic embeddings of words, allowing flexible semantic evaluation. We outline a number of hypotheticals in which the dependence of BertScore on token embedding cosine similarity may be exploited. The comparative distribution of BertScores on a set of reference - prediction pairs mean that results often scale differently with training to traditional metrics, which requires more expertise when interpreting results.
In this paper, we demonstrate an improvement to BertScore, using accelerated Linear Sum Assignment approximations that reduce the mean score while maintaining accuracy. Linear Sum Assignment allows BertScore to be more easily understood in the context of other NLG metrics, by changing the distribution of the metric.

Keywords: NLG Metrics · BertScore · Linear Sum Assignment

1 Introduction and Background

A variety of metrics are used for the evaluation of models performing NLG tasks. The most common of which is Bleu [10], a metric originally designed for machine translation but frequently used in machine translation, summarisation, and other NLG tasks. Other task-specific metrics are frequently used, such as Rouge [9] and Meteor [4] for summarisation. These metrics are largely based on comparing a machine generated prediction text to a human-written reference text, analysing the number of n-grams which match between the two texts. Although these metrics use differing operations to provide a final score between 0 and 1, representing the similarity between the texts, they are all limited by their

* corresponding author

reliance on n-gram matching. Where a machine may generate a sentence which uses multiple words that are synonyms to that which a human may write, all of these scores will be low, despite the sentiment of the sentences matching.

To address the issue of NLG metrics not recognising the sentiment of a sentence, Zhang et al. [13] propose BertScore, which is a metric used to measure the similarity between two pieces of text. BertScore uses embeddings from an LLM - the Bert (Bidirectional Encoder Representations from Transformers) model - to capture the sentiment of words within a text before using a cosine similarity matrix to compare texts. While this captures the sentiment of the texts being compared, there are flaws in the methods used.

BertScore may incorrectly give an overinflated score where repeated tokens have similar latent embeddings, especially if a similar latent embedding is in the other sequence accidentally. Such similarities may readily occur when decoding and reencoding between tokenization schemes. Many tokenizers have conflicting tokenization schemes, which may cause this. For example, the CLIP tokenizer [11] uses a padding token containing a zero, which clashes with the encoding of an exclamation mark, causing it to look identical to a model.

Here are the steps involved in calculating BertScore:

1. Tokenization: The input text is divided into individual tokens. Tokenization is done using the WordPiece tokenizer, which breaks words into subwords and assigns each subword a unique token.
2. Encoding: Each token is passed through a pre-trained model.
3. Similarity calculation: Encoded representations of the tokens are used to calculate the similarity between the two pieces of text. BertScore uses the cosine similarity metric to measure the similarity between the vectors. The maximum values are summed for each prediction and reference.
4. Aggregation: Finally, individual token similarities are aggregated to obtain an overall similarity score. BertScore uses a variant of the F1 score, called the Precision-Recall-F1 (PRF) score, to combine token similarities. The PRF score takes into account both precision (how many tokens are correctly matched) and recall (how many tokens are missed).

In this paper, we generate latent embeddings and the token similarity matrix, as in BertScore, and use it to compare 5 approximations of Linear Sum Assignment. Each approximiation is applied to the in generating the BertScore metric, in order to provide an improvement to the metric by reducing the mean score while maintaining accuracy, making the results better reflective of the texts being analysed.

2 Related Works

In 2020, BertScore [13] was introduced, which has since emerged as the predominant metric for NLG based on large language models. Other similar LLM-based metrics for NLG tasks include MoverScore [14], which focusses on the principle of measuring semantic distance to improve the metric; and FrugalScore [5], which

aims to minimise the environmental impact of BertScore while maintaining accuracy.

Since then, approaches like BartScore [12] have replicated the efficacy achieved using different underlying models. Fig. 5 demonstrates the importance of such work, showing that although we have used a range of different approaches, some models are simply not suitable for some metric styles, as shown by the use of the CLIPScore [6] methodology with the Albert model [8]. However $\text{Bart}_{\text{base}}$, a model which other metrics are using, performs lowest, illustrating that poor scores may just mean that the end of text token is treated differently in some models rather than being a primary indicator of poor performance.

CLIPScore makes use of the embedding of the final token in a sequence as a summary of the semantics of a set of tokens. CLIPScore explicitly uses a pretrained CLIP model for a metric grounded in a visual domain. We deviate from using the specific text encoder to instead apply the approach to the pretrained models being tested.

3 Methods

3.1 Proof of concept

Fig. 1 plots the distribution of sequence lengths within the EN-DE dataset from the larger WMT 2016 dataset [2]. In this work, two sequence lengths are tested: 128 and 384. Fig. 2 illustrates that padding to different lengths will significantly increase the frequency of padding tokens. Repetition of tokens will be the test used to illustrate how repeated tokens negatively affect BertScore metric. For this purpose, we disable the masking of padding tokens during calculation. This allows us to simulate the cases where sequences have many repeated tokens without having to generate out-of-distribution inputs.

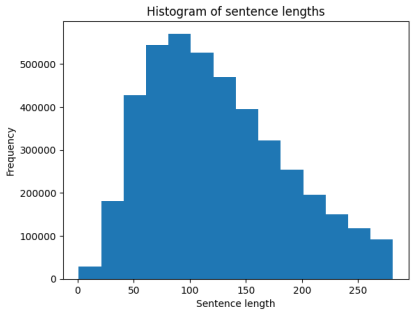

Fig. 1. A Histogram showing sequence lengths in WMT 2016 dataset. From this we derive using sequence lengths of 128 and 384 as test cases for how padding tokens skew BertScore

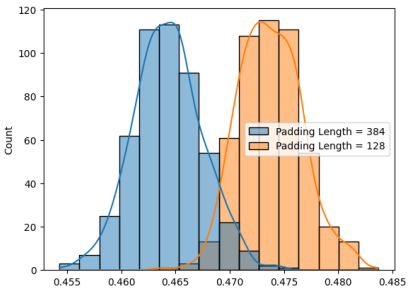

Fig. 2.
A showing the scores of DistilBert in a perfect case, with the distributions of padding length 128 and 384 imposed over each other

3.2 Linear Sum Assignment

The Linear Sum Assignment(LSA) problem is a combinatorial optimisation problem that deals with finding the best assignment of a set of tasks to a set of agents in such a way that the total cost or benefit is minimised or maximised. The problem can be mathematically formulated as follows:

Given a similarity matrix S, where S_{ij} represents the similarity between the ith token in a reference sequence and the jth token in a prediction, the goal is to find a permutation matrix P that maximises the objective function:

$$\text{maximize} \sum_{i=1}^{n} \sum_{j=1}^{n} S_{ij} P_{ij}$$

where n is the number of assignments to make, which is equal to the minimum of matrix width and matrix height. Subject to the constraints that each task is assigned to exactly one agent, and each agent is assigned to exactly one task. The rules that govern Linear Sum Assignment are as follows:

1. Each row and column of the matrix must contain exactly one assigned element to ensure that each resource is assigned to exactly one task and vice versa.
2. No two assigned elements can be in the same row or column, to prevent multiple resources from being assigned to the same task or multiple tasks from being assigned to the same resource.
3. The assignment should be made in such a way that the sum of the assigned elements is minimised or maximised, depending on the objective that is defined based on the problem at hand.

To solve the Linear Sum Assignment problem, various algorithms can be used, such as the Hungarian algorithm [7] or the auction algorithm [1]. These algorithms efficiently find the optimal assignment by iteratively updating the assignment matrix until the optimal solution is reached. However, in common libraries, these methods are regularly CPU bound and can add an unreasonable overhead if the accelerator has a high transfer latency.

In this paper, we have chosen the approximations detailed in Table 1 due to the widespread use of accelerators in contemporary machine learning workflows. Our source code for comparing these approximations is available online. [1]

We detail whether these algorithms are suitable for gradient calculation. The approaches employed in this study do not currently incorporate a gradient; as we discuss in Section 6, integrating gradients is an avenue of future work that we intend to consider. Instead of removing nonzero elements with a one-hot mask, approaches like using Gumbel-Softmax could be used to maintain gradients while yielding a one-hot output. This creates potential for use in a wider host of ML applications.

Table 1. Comparison of Methods.

Method Name	Accelerator	Can have gradient	Descriptor
Stock	✗	✗	The hungarian algorithm implemented in DETR-style computer vision [3] applications found in the scipy optimise library.
Row masking approach	✓	✗	A rework of the stock algorithm in Pytorch that works by calculating the deltas between the top values in each column of the tensor. Select the one with the largest delta and mask the row corresponding to the highest value. Repeat until all values are assigned.
Recursive approach with Argmax	✓	✗	For a fixed number of iterations (optimally around 6-8) add the delta of each value to the highest other value in the row and column, an auction style method that can maintain a gradient. Using argmax to select the result.
Recursive approach with TopK Filtering	✓	✓	Use "topk" to choose the best configuration after the above steps. No guaranteed compliance with LSA rules.
Aggressive Auction with TopK Filtering	✓	✓	A more ruthlessly efficient version of the initial recursive method with the same argmax column selection.

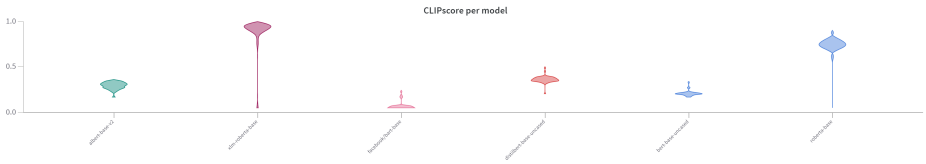

Fig. 3. The distribution of CLIPScores per model. To be used as baseline evaluation scores. Independent of the LSA algorithm and padding.

4 Results

To fully evaluate this effect, the baseline of CLIPScore [6] is used: the cosine similarity of the latent embeddings of the "[EOT]" token in each sequence. This score is not affected by LSA or the length of the sequence. It can be concluded from Fig. 3, that the selected models have a wide range of performance profiles in the chosen dataset, which is an indicator that our proposed method applies evenly across the distribution. From Fig. 3, each model has a very unique

[1] A replication package containing the code we have used throughout this paper can be found at anonymous.4open.science/r/LiSAScore-9330.

performance that does not change with other factors. The expectation is therefore that LiSAscore created a figure with sufficient fidelity to be able to identify a model by performance. We would therefore expect the violin plot of each LSA approach to have many individual peaks in the best case. As each plot represents the outputs from all models and each has a unique profile in this domain, the models' performance should be visible within the plot of all runs.

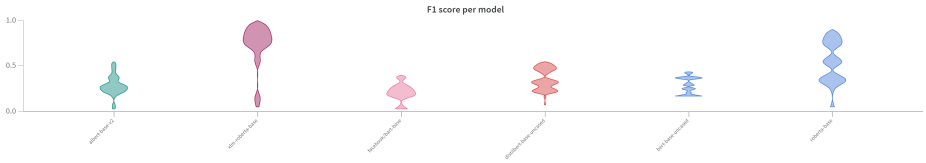

Fig. 4. F1 Scores grouped my model used across all runs.

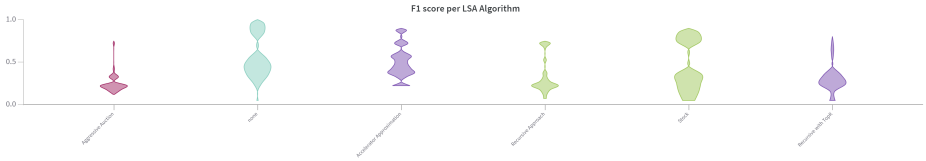

Fig. 5. F1 scores across all runs grouped by algorithm. Notable for numbers of visible peaks in each violin-plot

When comparing Fig. 5 to Fig. 4, with the baseline in Fig. 3, we conclude that the use of LSA is much more granular and sensitive to hyperparameters. LSA algorithm "Accelerator Approximation", which represents the accelerator-based row masking approach as detailed in Table 1, shows five distinct and evenly spaced regions, which correspond to the models CLIPScores. It can be seen in Fig. 5, that runs without LSA have comparatively few peaks and troughs, meaning that it is not as good an indicator of performance as CLIPScore and may be overly sensitive to many tokens. A key observation of Fig. 5 is that recursive algorithms accentuate the distinction between good and bad, but appear to be poor representations of which model was used. As improving interpretability is a key goal, this makes the recursive family of approximations offer a significant advancement: Allowing the optimum case to be isolated irrespective of performance. The increased distance and reduced variance in each peak is alteration of the score matrix as a solution is iterated toward, which is demonstrated in the following example in Fig. 7 and Fig. 6. This phenomenon is caused by recursive approximations assigning extreme values rather than strict adherence to the governing principles of LSA. The magnitude increase of values used in the set of recursive algorithms

is demonstrably effective for evaluation metrics based on the distance between good and bad samples.

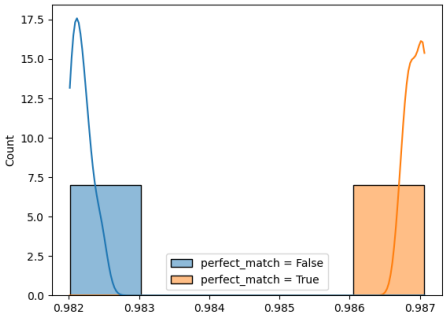

Fig. 6. The F1 scores of positive and in-batch negative sequence generated with XLM-RoBERTa without LSA

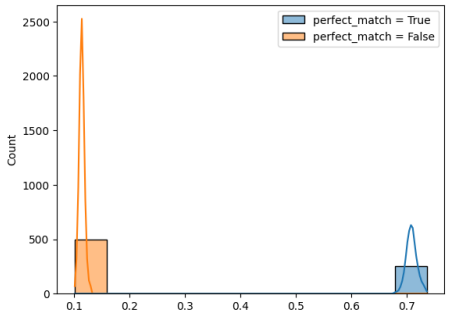

Fig. 7. The comparative F1 scores of positive and in-batch negative sequence generated with XLM-RoBERTa with a recursive LSA approach

An initial comparison of Fig. 6 and Fig. 7 shows the proportional difference in the resultant scores produced. The new results are significantly more meaningful to users, demonstrated using the recursive function with Argmax for LSA. Taking into account the inclusion of the data in Fig. 7, this shows that our improvement is well emphasised by the difference in model performance. In the baseline of Fig. 6, we can show that the difference between sequences and in-batch negatives is very minimal using BertScore. By contrast, using Linear Sum Assignment, although both scores are reduced, the difference between positive and negative samples is significantly more pronounced, both in absolute and relative terms.

5 Conclusion

We have shown that implementing Linear Sum Assignment in BertScore in place of a maximal value selection improves usability as a metric, as demonstrated across many pre-trained models. Our approach minimises the risks and errors posed by repeated tokens in the input or prediction sequence.

We show that using the Linear Sum Assignment algorithms provides a significant increase in usability and conformity with other metrics. We have demonstrated that using Linear Sum Assignment increases the metric distance between different perturbations sufficiently so that individual permutations form distinct clusters. We have also outlined how different approximations using recursive algorithms behave differently for this usage, which may have significant implications for future work. From this work, there is a clear recommendation that recursive approximations are used to emphasise the correct or incorrect behaviour of a

model, where stock implementations and nearby approximations offer a more stable metric for all use cases.

We recommend further exploration of the comparative strength of LiSAScore compared to other model-based scoring with respect to known edge cases. We consider this work to be foundational for further exploration of these metrics as part of gradient descent training.

6 Future Works

Although this paper primarily provides an improvement in the metric, the greatest result is a family of algorithms that can now maintain a gradient.

At current, to perform a contrastive loss, the padding tokens must be kept, to maintain equal sequence lengths, and masked. Whether they are masked or not, they are likely to dominate the result of any contrastive loss. The concept of contrastive loss here is juxtaposed with BertScore's objective is to have a metric that is tolerant to deviating word order. More research is required on the performance of LiSAScore within summarization tasks, where sequence lengths may have significant imbalance. In such a case, LiSAScore forces only the minimum sequence length of tokens to be selected, which may have a positive effect on the scores and punish excessively short summaries. Without LSA, the selection of tokens in the output sequence is predominantly biased to match against every token in the input sequence. The frequency of stop words and vague statements can be gamified to fool these metrics. By using Linear Sum Assignment with other methods like IDF scaling, the counting imbalance between the two sequences of differing lengths can be minimised.

Limitations

During the implementation of these methods, we iterate over the batch dimension, using a simple for loop. There is much potential to massively reduce the cost of this by batching the operations, as most take place over a fixed step count, and further optimisations are readily findable. Such optimisations are regrettably outside the scope of these findings and can be implemented upon acceptance of this concept.

We also acknowledge the limitations of storage space and available hardware for the tests, which meant replication across other datasets and tasks has been limited, although rudimentary experimentation suggests that other datasets would offer concurrent results.

Ethics Statement

The primary ethical considerations for this document are the environmental impact of the use of LLMs and the provenance of the data we have used. Although any use of LLMs has a measurable environmental impact, we have made every

effort to reduce this impact. In total, we conducted 336 runs of our experiment, totalling 17 days of compute time. To ensure that our experiment is as replicable as possible, we used the EN-DE dataset from the larger WMT 2016 dataset [2] - which contains well-documented and publicly available data.

Acknowledgements

We would like to express our gratitude to the HPC offerings at Lancaster University, especially the multi-node cluster of the UCREL NLP-Group.

Bibliography

[1] Bertsekas, D.: New auction algorithms for the assignment problem and extensions. Results in Control and Optimization **14**, 100383 (2024)

[2] Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Jimeno Yepes, A., Koehn, P., Logacheva, V., Monz, C., Negri, M., Névéol, A., Neves, M., Popel, M., Post, M., Rubino, R., Scarton, C., Specia, L., Turchi, M., Verspoor, K., Zampieri, M.: Findings of the 2016 conference on machine translation. In: Bojar, O., Buck, C., Chatterjee, R., Federmann, C., Guillou, L., Haddow, B., Huck, M., Yepes, A.J., Névéol, A., Neves, M., Pecina, P., Popel, M., Koehn, P., Monz, C., Negri, M., Post, M., Specia, L., Verspoor, K., Tiedemann, J., Turchi, M. (eds.) Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers. pp. 131–198. Association for Computational Linguistics, Berlin, Germany (Aug 2016). https://doi.org/10.18653/v1/W16-2301, https://aclanthology.org/W16-2301

[3] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European conference on computer vision. pp. 213–229. Springer (2020)

[4] Denkowski, M., Lavie, A.: Meteor universal: Language specific translation evaluation for any target language. In: Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Monz, C., Post, M., Specia, L. (eds.) Proceedings of the Ninth Workshop on Statistical Machine Translation. pp. 376–380. Association for Computational Linguistics, Baltimore, Maryland, USA (Jun 2014). https://doi.org/10.3115/v1/W14-3348, https://aclanthology.org/W14-3348

[5] Eddine, M.K., Shang, G., Tixier, A.J.P., Vazirgiannis, M.: Frugalscore: Learning cheaper, lighter and faster evaluation metricsfor automatic text generation. arXiv preprint arXiv:2110.08559 (2021)

[6] Hessel, J., Holtzman, A., Forbes, M., Bras, R.L., Choi, Y.: Clipscore: A reference-free evaluation metric for image captioning. In: EMNLP (2021)

[7] Kuhn, H.W.: The hungarian method for the assignment problem. Naval research logistics quarterly **2**(1-2), 83–97 (1955)

[8] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942 (2019)

[9] Lin, C.Y.: ROUGE: A package for automatic evaluation of summaries. In: Text Summarization Branches Out. pp. 74–81. Association for Computational Linguistics, Barcelona, Spain (Jul 2004), https://aclanthology.org/W04-1013

[10] Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Isabelle, P., Charniak, E., Lin, D. (eds.) Proceedings of the 40th Annual Meeting of

the Association for Computational Linguistics. pp. 311–318. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA (Jul 2002). https://doi.org/10.3115/1073083.1073135, https://aclanthology.org/P02-1040

[11] Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)

[12] Yuan, W., Neubig, G., Liu, P.: Bartscore: Evaluating generated text as text generation. Advances in Neural Information Processing Systems **34**, 27263–27277 (2021)

[13] Zhang, T., Kishore, V., Wu, F., Weinberger, K.Q., Artzi, Y.: Bertscore: Evaluating text generation with bert. In: International Conference on Learning Representations (2020)

[14] Zhao, W., Peyrard, M., Liu, F., Gao, Y., Meyer, C.M., Eger, S.: Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance. arXiv preprint arXiv:1909.02622 (2019)