Rethinking Text Generation Evaluation: A Unified Evaluation Theory for Reflective and Open-Ended Generation Tasks

Anonymous ACL submission

Abstract

With increased accessibility of machine generated texts, the need for their evaluation has also increased. There are two types of text generation task for which evaluation is required. In open-ended generation tasks (OGTs), the model generates de novo text without any input on which to base it. Examples include story generation. In reflective generation tasks (RGTs), the model output is generated to reflect an input sequence. An example is machine translation. Evaluation of RGTs is well-researched, and typically uses metrics that compare one or more gold-standard references to the model output. Evaluation of OGTs is less well-researched, and reference-based evaluations are more challenging: as the task is not seeking to reflect an input, there are usually no references. In this paper, we propose a theory of evaluation that covers both RGT and OGT evaluation. Based on this theory, we propose an output-oriented reference generation method for OGTs, develop an automatic language quality evaluation method for OGTs, and review previous literature from this new perspective. Our experiments demonstrate the effectiveness of these methods across informal texts, formal texts, and domain-specific texts. We conduct a meta-evaluation to compare existing and proposed metrics, finding that our approach better aligns with human judgement.

1 Introduction

Natural language generation (NLG) has progressed significantly in the last decade. This progress has been made through the use of encoder-decoder (Lewis et al., 2019) and decoder only architectures (Brown et al., 2020; Touvron et al., 2023). In the last few years, the use of these transformer-based architectures (Vaswani et al., 2017) and increased compute capacity to create generative Large Language Models (LLMs) such as Brown et al. (2020); Touvron et al. (2023) has attracted attention from both academia and the public. However, the lack of good evaluation metrics for generated text has limited the ability to make informed choices of the best machine generated candidates from one or multiple LLMs.

NLG tasks can be categorised into one of two types: reflective generation tasks (RGTs) and open-ended generation tasks (OGTs). In RGTs, the model output is a reflection of the information in the input. The output is restricted by the input, and its content must be faithful to the input. Such tasks include machine translation and summarisation. OGTs generate new information that does not exist in the input. Examples of such tasks include story generation and synthetic medical report generation.

Many studies (Sellam et al., 2020; Zhang et al., 2019; Papineni et al., 2002; Rei et al., 2020; Stanojević and Sima'an, 2014; Banerjee and Lavie, 2005) on RGT evaluation focus on comparing the similarity between pre-written human references and machine-generated outputs. However, these methods often consider only the similarity metric used and overlook the choice of references, which may not necessarily give an accurate final evaluation of the synthetic text quality. OGT evaluation is a less researched area, due to the difficulty of creating pre-written human references (Yue et al., 2022). Much research on OGT evaluation has instead compared the distributional similarity between corpora of synthetic text and corpora of real text in the target domain, using for example statistical methods such as perplexity (Bhandari et al., 2020) or self-BLEU (Zhu et al., 2018) to measure this similarity. Other researchers such as Pillutla et al. (2021) estimate the underlying model distribution from the corpora and measure the distance between this and the real text distribution using Kullback-Leibler (KL) divergence.

These evaluation approaches have two major problems: (1) in OGT evaluation, they are unable to provide a measure of the text quality of each individual text; (2) there is no common conceptual framework or way of communicating and comparing evaluation metrics between these two text generation paradigms. This prevents us from using the more thoroughly researched RGT evaluation methods in OGT and drawing useful conclusions across the two tasks.

This paper provides such a common conceptual framework, bridging between evaluations of RGT and OGT. Based on this framework, we develop a new evaluation method for OGTs that assesses individual text quality without recourse to any reference. We call this evaluation method **ARGENT** (**A**utomated **R**eference-free **E**valuation of **GEN**erated **T**ext). In order to compare the existing OGT evaluation metrics, such as Self-BLEU and Mauve, with our own metrics, we develop a comparison between evaluation metrics, i.e. an evaluation of the evaluation metrics, which we refer to as a meta-evaluation.

The contributions of this paper are:

- A unified theoretical view of generative text evaluation.
- ARGENT: a reference-free solution for automatic OGT language quality evaluation.
- Comparisons of evaluation metrics, i.e. meta-evaluations, with different text types.
- An output-oriented reference generation method for OGTs.
- A short review of the literature on generative model evaluation, from a new perspective.

2 Unified Theory for Generative Task Evaluation

To illustrate the importance of reference choice in evaluating generative tasks, we consider the following simple task, translation of the French sentence "C'est vraiment un homme intelligent" into English. Let us assume that we are comparing two models. Model 1 output is "He truly a smart man". This is largely correct, but missing the verb. Model 2 output is "He truly is a clever dog", with the noun completely wrong. Table 1 lists a set of possible correct translations (references) and the scores from different metrics comparing the outputs against these references. From the table, we can see: 1) Evaluation metrics can vary significantly based on the references used. If the last reference is used for evaluation, then with all three metrics, "He truly is a clever dog" will be picked as a better answer. 2) With BERTScore, the differences between references are smaller than with BLEU and ROUGE. This demonstrates that better metrics, such as those that take in to account semantics, can reduce variability caused by different references and thus may alleviate the problems caused by these.

References	**BLEU**	**ROUGE-L**	**BERTScore**
Candidate 1:**He truly a smart man**			
He truly is a smart man	82.24	90.91	96.14
He really is a smart guy	45.42	54.55	93.62
He really is an intelligent guy	18.18	0.50	93.30
He truly is a clever man	49.45	72.73	94.98
Candidate 2: **He truly is a clever dog**			
He truly is a smart man	55.68	66.67	94.72
He really is a smart guy	37.95	50.00	92.98
He really is an intelligent guy	26.04	33.33	92.62
He truly is a clever man	82.94	83.33	95.45

Table 1: Scores of two translation candidates against different references with different metrics

Evaluating language generation is very different from evaluating traditional classification and regression tasks. This is because language generation does not have a finite list of possible output classes, as is found with classification: in the translation example above, there are multiple possible correct outputs. Additionally, language generation does not have a straightforward measurable scale of output like the continuous numerical scale used in regression. Thus, one cannot measure the performance directly against the references. In addition, most language generation tasks do not have one correct answer, with many not even having a finite list of acceptable answers.

In any evaluation of a text generation model, we have:

- **Output** - the text generated by the model, e.g. the candidate translation in a machine translation task.
- **Reference space** - the set of all possible gold-standard references, or possible ground truth texts. These are all texts that are correct answers to the generation problem. In a machine translation task, these would be all possible correct translations. In a synthetic document generation task, they would be all possible correct documents.
- **Reference** - a single text sampled from the reference space.
- **Similarity score** - a measure of the similarity between the output and a reference.

Let us use **Y** to denote the set of all the possible gold-standard references, and $\hat{Y}$ to denote the output of the model. The evaluation of output $\hat{Y}$ can be defined as

$$E = max(f_{similarity}(\hat{Y}, Y_i), \forall Y_i \in \mathbf{Y}) \qquad (1)$$

where $f_{similarity}$ is the similarity score function used by the evaluation metric, e.g. BLEU, ROUGE, BERTscore.

Key points to consider with respect to this definition are:

- In some literature, evaluation and similarity scores are conflated. However, the ability of an evaluation to differentiate between two models also depends on the references used, as demonstrated in the machine translation example above. In this paper, we use the term evaluation to refer to the combination of the way in which references are selected, and the similarity score used.
- For any similarity score in a given evaluation, there exists a set of possible answers, one between the model output and each possible reference. The similarity score for the true evaluation of this output is defined by the maximum, i.e. the score between the output and the most similar reference.
- Some similarity scores are better than others. For example, a score which takes in to account the grammatical structure and semantics of a text will be better than one which only takes into account word frequency. The best possible, or perfect, similarity score is the one that comes closest to human judgement of the similarity between two texts.
- This definition covers both reflective generation and open ended generation. The difference between them is the size of the reference space. Reflective generation has a restricted reference space while open ended generation has more flexibility, giving a larger reference space.

We further illustrate the effects of references and similarity metrics in Appendix A.

3 Auto-Evaluation for Language Quality

The large reference space in evaluation of open-ended generation leads to a problem. How do we find the closest reference? One solution is to use output-oriented human annotation in which a human judge corrects errors in an output by making the minimum number of changes, to give an error-free text that is the closest reference to the output.

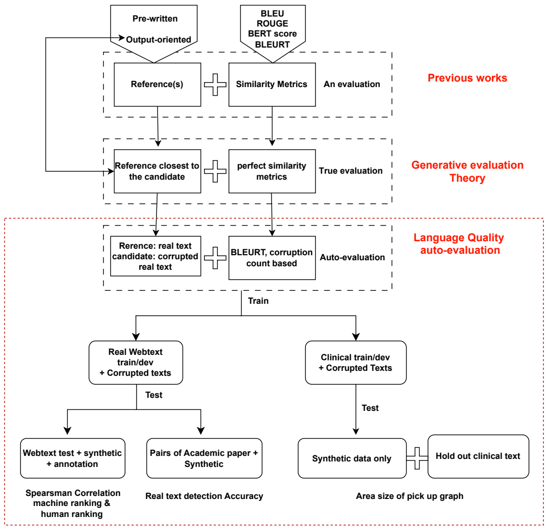


Figure 1: Relationships between different evaluation methods and experimental work presented in this paper

The output-reference pair can then be used in an evaluation. This technique has been applied for RGTs, such as machine translation, where it has been shown to gives scores more aligned with human judgement than pre-written references with a translation edit rate metric (Snover et al., 2006). Our unified evaluation theory suggests that a similar technique could also be used in OGTs, and that such an output-oriented reference annotation method could provide more accurate evaluations.

Such output-oriented evaluation is, however, expensive and does not scale. We could overcome this with an automated evaluation, but auto-evaluation may itself vary in quality, with some methods providing results that are closer to those of a perfect evaluation than others. We therefore need to consider ways in which we might measure the quality of auto-evaluations. The remainder of this paper discusses a new reference-free auto-evaluation method, ARGENT, and meta-evaluations of this and existing methods under different dataset conditions. Figure 1 shows the relationships between evaluation, generative evaluation theory, auto-evaluation, and the experiments reported in this paper.

3.1 ARGENT : Pre-trained Auto-evaluation on Corrupted Texts

To understand auto-evaluation, consider formula 1 as an evaluation model. Given a set of all possible references, and the output from some NLP generative model, this evaluation model will provide an evaluation score. However, it is not usually possible to create a set of all possible references. It is also hard to directly work out which reference, from all the possible references, gives the maximum score for a given output.

If, for some NLP task, we were able to create a set of proxies for model outputs, and if we know the evaluation score for these proxies in advance, we can envisage training an evaluation model on these proxies and their scores. This model would then be able to predict the evaluation score on previously unseen output for the same NLP task. Once trained, this evaluation model - ARGENT - would be able to predict an evaluation score without having seeing any reference. In order to create such a set of proxy outputs and their evaluation scores, we reverse reference generation. Rather than generate a reference for an output, we generate a likely output from a real text reference, by corrupting the real text in some way. This will give us a proxy model output paired with a reference which approximates the output. Moreover, the degree to which the reference approximates the output will depend on the amount of corruption, and can therefore be varied and quantified, providing a metric that describes how well the proxy output matches the reference, i.e. an evaluation score for the proxy output.

Text corruption Text corruption methods need to align with variation in language quality in generated text. In this regard, we propose two text corruption methods, an inflection method and a local shuffling method.

In the inflection method, the tokens in each sequence are inflected to different part-of-speech (POS) forms. For example, in the sentence "I like books", the token "books" is a plural noun. We can inflect it to a past tense verb "booked" to create the corrupted sentence "I like booked". In the work described in this paper, we use SpaCY POS tags[1] and we use the tagger module[1] and lemminflecction module[2] from SpaCy. In some cases, it is not possible to inflect a word. To overcome this, we restrict the tokens that are considered in this process to have POS tags in the list[3].

In the local shuffling method, we slide a window of variable length across the text and shuffle the tokens within this window. The window length is drawn randomly from a given range. When corruption and shuffling are both performed on the same text, we refer to this as shufflection.

The pseudo-code for inflection and local shuffling of a single report can be found in Appendix B, Algorithms 1 and 2. To generate a dataset with varying quality, the corruption rate is varied for each report in the dataset. In the experiments reported, the probabilities for corruption of each report are drawn randomly from a pre-defined range. The pseudo code for this process can be found in Appendix B, Algorithm 3.

Score generation We explore two methods for generating scores for corrupted output texts. In the first, the corruption score is calculated from the proportion of the total number of corruptions made across all corruption processes. For text length N, number of corruption methods K, and original token state k=0, the corruption score and text quality score is defined as:

$$S_{corruption} = \sum_{k=1}^{K} \sum_{i=1}^{N} (x_i^k \neq x_i^{k-1})/KN \qquad (2)$$

$$S_{quality} = 1 - S_{corruption} \qquad (3)$$

The second method is based on the BLEURT score, a state-of-the-art metric for comparing candidates and references in machine translation, which is trained on human judgements, and which uses context embeddings (Sellam et al., 2020). In ARGENT, we use BLEURT to assign a score to each reference paired with its corrupted proxy output. In both the BELURT based and corruption count based scoring methods, we use the score as the label when training the auto-evaluation model on the proxy outputs.

3.2 Meta-evaluation of evaluation models

For text generation datasets with human annotation, we can use the correlation between auto-evaluation and human evaluation to measure the performance of auto-evaluation models. Human annotation is, however, a difficult task that can result in inconsistent data (Clark et al., 2021; Karpinska et al., 2021). Given that synthetic text generators are trained on real data, with an objective to mimic real data, it can be assumed that the language quality score of a real text should be no less than that of the synthetic text. With this assumption, we can build test tasks without human annotation.

In some limited text generation cases, a set of pairwise real and synthetic texts do exist. For ex-

[1] https://spacy.io/api/tagger

[2] https://spacy.io/universe/project/lemminflect

[3] JJ, JJR, JJS, NN, NNS, NNP, NNPS, RB, RBR, RBS, VB, VBD, VBG, VBN, VBP and VBZ

ample, Liyanage et al. (2022) pairs real texts with versions in which a few sentences are substituted by generated texts. These are used to train generated text detection models. In evaluation, model scores between the real and these semi-synthetic texts are compared. A true positive exists if the real text score is greater than that of the semi-synthetic text score.

For cases in which no such pairs exist, we propose a batch level approach. A batch of of texts, say 100, are selected, among which 90% are synthetic and 10% are real. All texts in the batch are ranked by their auto-evaluation scores. The top k% of ranked texts are then sampled, with k varying from 1 to 100. For each k, the number of real texts found in this top k% is calculated, as a percentage of the total number of real texts. We refer to this as the pick-up rate, i.e. the rate at which the auto-evaluation is able to pick the real texts. An example pick-up rate graph is shown in Figure 2, where the x axis gives the top k% samples of the ranking, and the y axis gives the pick-up rate of real texts among the top k% samples. For a 90% to 10% split of synthetic to real texts, the best case is when all real texts are placed in the top 10% of the ranking, which corresponds to the upper bound line in the graph. In the worst case, all real texts would be placed in the bottom 10% of the ranking, which is shown by the lower bound line. If we were to rank the texts randomly, there is a probability that 10% of real texts would be picked up at every decile, which is represented by the diagonal line in the graph. For an auto-evaluation model, the area between its curve and the lower bound reflects how good the auto-evaluation model is. We define a performance metric, given by the area under the model curve as a percentage of the area between the upper and lower bounds. As the model curve is discrete from 0 to 100, the area is calculated by summation of the height above the lower bound line at each discrete point. The diagonal random ranking line defines an area half that between the bounds, and therefore an evaluation score of 50%.

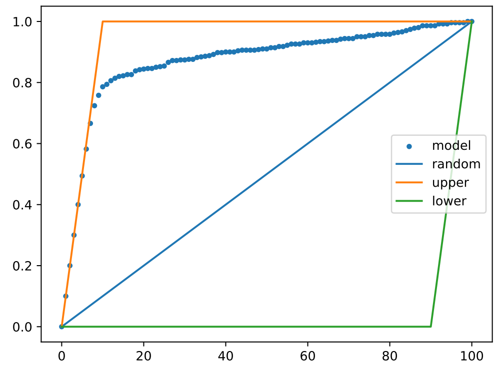


Figure 2: Example pick-up rate graph

4 Experiments

Data and metrics To test our theory, we carried out experiments on three different type of texts: formal, informal and domain-specific. The details of datasets used for each type can be found in the corresponding subsections below. We report correlation, accuracy and pick-up graph area for different tasks and datasets as discussed below.

Auto-evaluation models: Unless specified otherwise, all auto-evaluation ARGENT models reported in this paper are based on BERT-base cased (12 layers, 768 hidden units, 12 heads) (Devlin et al., 2018). We pre-train ARGENT models on corrupted texts and deploy on test tasks that consists of either machine generated text or real text without fine-tuning on this test data. For pre-training, we use batch size 32, learning rate 1e-5, and 3 training epochs. The model has about 110M parameters, and was trained on a single A100 GPU.

Pre-training dataset: Unless specified otherwise, all pre-training datasets are built using inflection and shuffling on real text. We conducted grid search of the inflection and shuffling probability ranges of 0.2, 0.4, 0.6, 0.8, 1.0 for each pre-training dataset, and we use the combination of the two best performing probability ranges of each method for shufflection. The scores for each corrupted text are calculated using both corruption count based scoring and BLEURT based scoring.

4.1 Informal Text Evaluation: WebText

Dataset and Metrics Evaluation of informal text used the WebText dataset [4]. For ARGENT training, we use the training and validation data splits provided in WebText. We use the WebText test data previously annotated and reported with Mauve, which includes synthetic data generated by eight different generative models (Pillutla et al., 2021). In this test dataset, the annotation is done by pair-

[4] https://github.com/openai/gpt-2-output-dataset

wise text comparison on three criteria: human-like, sensible, and interesting. The pairwise preference annotations are then aggregated into a ranking by fitting a Bradley-Terry (BT) model to the output from the eight generative models. (Marden, 1996; Pillutla et al., 2021).

We test ARGENT models across texts generated by all eight generative models, as provided in the Mauve test set. In order to provide a score comparable to those reported in Pillutla et al. (2021), we create an auto-evaluation ranking by averaging the scores assigned to texts generated from each of the eight models. We then calculate the Spearman rank correlation between the human judgements provided by the test set and our machine rankings, ranging from -1 to 1, with a higher positive value indicating stronger positive correlation, as is used in the Mauve paper (Pillutla et al., 2021). However, we need to treat this performance metric with caution, because the correlation is based on the ranking of only eight generative models, an insufficient sample size to give a reliable correlation.

Results Table 2 compares the Spearman correlations of ARGENT to those from six previously published evaluation models. We report the best performing ARGENT model, which is based on shuffling with probability range 0-0.8 and count-based score (see Appendix C Table 5 for performance of other models). From the results, we can see that ARGENT achieved the second-best performance for every criteria, just behind the Mauve model. Mauve, however, has two drawbacks compared to our auto-evaluation model. First, it requires a human-generated corpus. Second, it creates a single score for the model generating the test corpus, whereas ARGENT is creating an individual score for each report in that corpus, which we have averaged for the purpose of comparison to Mauve. The Sensible criterion is the closest criterion to language quality evaluation, on which ARGENT is comparable to Mauve. The Human-like criterion can also reflect language quality. Mauve benefits from directly measuring the distribution similarity between human text and machine generated text, whereas ARGENT, as a zero-shot learning model, is trained on corrupted data that is different from the synthetic data used for testing.

4.2 Formal Text Evaluation: Synthetic Academic Publications

Data and Metrics We use the fully generated academic papers dataset from Liyanage et al. (2022) to evaluate performance on formal text. There are 100 papers in the corpus. We provide comparisons between ARGENT, trained on WebText data, to other models reported in Liyanage et al. (2022), including results for BERT-based models trained on news headlines (Brown et al., 2020) . The use of an auto-evaluation model trained on WebText data to evaluate a very different type of text illustrates ARGENT's ability to adapt to different types of text.

Results The best result was achieved by ARGENT, using inflection with probability 0-0.6 and BLEURT scoring. This is shown in Table 3 along with those of other studies in the literature. Results for other configurations of our method are given in Appendix D Table 6.

Model	Accuracy
Bag of ngrams 1-3, MNBA (1)	19.7
Bag of ngrams 1-3, PACA (2)	31.8
Bag of ngrams 1-3, MCH (3)	19.7
Bag of ngrams 1-3, SVM (4)	39.7
LSTM model (Maronikolakis et al., 2020)	59.1
Bi-LSTM (Maronikolakis et al., 2020)	40.9
BERT (Maronikolakis et al., 2020)	52.5
DistillBERT (Maronikolakis et al., 2020)	62.5
ARGENT	**97.0**

Table 3: Performance of different evaluation models on academic publications. Liyanage et al. (2022) used Bag of ngrams as features for (1) MNBA - Multinomial Naive Bayes Algorithm (2) PACA - Passive Aggressive Classifier Algorithm (3) MCH - Multinomial Classifier with Hyperparameter (4) SVM - Support Vector Machine

4.3 Domain-specific Text Evaluation: Clinical Text

Data and Metrics To test ARGENT performance on domain-specific text, we generated synthetic reports using BioGPT (Luo et al., 2022) trained on clinical reports from a large secondary healthcare provider (this work is currently under review). We have chosen to use clinical text because real texts are often difficult to obtain in a healthcare setting, for privacy and ethical reasons. Synthetic clinical text can therefore be useful for NLP development, pre-training, and in education. We generated 97152 reports, with 92652 used for training and 4500 held back for testing. There are five types of clinical reports. Details of these types and the training and validation splits can be found in Appendix E Table

Metric	Gen. PPL	Zipf Coef.	REP	Distinct-4	Self-BLEU	Mauve	ARGENT
Human-like	81.0	83.3	-16.7	73.8	59.5	**95.2**	85.7
Sensible	73.8	69.0	-7.10	59.5	52.4	**85.7**	81.0
Interesting	64.3	52.4	-14.3	52.4	40.5	**81.0**	73.8

Table 2: Performance of different evaluation models on WebText (1) Generative perplexity (Fan et al., 2018) (2) Zipf Coefficient (Holtzman et al., 2019) (3) Repetition (Pillutla et al., 2021) (4) Distinct 4 n-grams (Pillutla et al., 2021) (5) Self-BLEU (Zhu et al., 2018) (6) auve (Pillutla et al., 2021)

7. For testing, we calculated the area size of pick-up rate graphs on 10 different sets of reports for each type, each set consisting 10 real reports and 90 synthetic reports. We report overall performance here. Results for individual report types are given in Appendix E.

Results The grid search of probability ranges for each evaluation method can be found in Appendix E Table 8. For the inflection with count-based score, the best probability range is 0-0.4; for inflection with BLEURT scoring, the best probability range 0-1.0; shuffling count based, 0-0.4; shuffling BLEURT-based, 0-1.0; shuffection count-based, shuffling 0-0.6 and inflection 0-1.0; shuffection BLEURT-based, shuffling 0-0.8 and inflection 0-1.0. Table 4 shows the best overall results for each ARGENT model. The best performing model is the shuffling model with a count based score, at 79.3% (>50%). This experiment shows that ARGENT can be effectively used in this domain-specific setting.

ARGENT models	Score
Inflection_count	68.1±2.4
shuffling_count	**79.3±2.6**
shuffection_count	67.7±3.5
Inflection_bleurt	58.7±5.8
shuffling_bleurt	56.8±6.4
shuffection_bleurt	59.4±6.1

Table 4: Performance of different ARGENT auto-evaluation models on clinical reports

5 Literature Review

In previous reviews of evaluation research such as (Zhou et al., 2023)(Yuan et al., 2021), evaluation has been categorised based on task type and evaluation method. For example, (Zhou et al., 2023) reviewed work based on the input and output type of the task, while (Yuan et al., 2021) classified evaluation methods into supervised, unsupervised and automatic metrics. In this work, we review the main evaluation methods described in the literature along the two dimensions of our evaluation theory: how the references are selected, and how the similarity score is defined.

5.1 Gold-standard reference selection

There are generally two types of references in RGT evaluation: pre-written human references and output-oriented references.

Pre-written references Most studies use pre-written human references, often using multiple references to reduce inaccuracy. Many shared-task evaluation datasets provide such references. For example, the WMT dataset[5], a widely-used machine translation evaluation benchmark, provides a set of gold standard references for each translation task, which is used by studies such as BERTScore(Zhang et al., 2019), BLEURT(Sellam et al., 2020) and BartScore(Yuan et al., 2021). There is little research on justifying pre-written reference selection.

Output-oriented References Some studies use output-oriented references, which may be referred to as human-in-the-loop or human-targeted references(Snover et al., 2006). For example, in Snover et al. (2006), references are made by manually editing the model output until it is fluent and has the same meaning as the input sentence. A similarity scores is calculated on these human-corrected references and on pre-written references using Translation Edit Rate (TRE) (Przybocki et al., 2006), BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005) metrics. Scores when using human-targeted references shows higher correlation with human judgement for all three metrics. This is in line with our unified generative evaluation theory. As far as we are aware, application to OGTs has not been discussed in the literature

[5]https://www.statmt.org/wmt22/metrics/index.html

5.2 Similarity Metrics

There are far more studies on similarity metrics, both supervised by training on human judgement as a regression problem, and unsupervised when based on matching or overlapping between synthetic text and references. Features used in the metrics may be statistical or embedding based.

Unsupervised metrics For statistical based features, BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004) compare text similarity by counting overlapping n-grams. Przybocki et al. (2006) uses edit distance to measure the dissimilarity between the output and the reference. Embedding-based unsupervised metrics, on the other hand, use neural networks to embed texts into vector space and compare the similarity of these vectors between output sequences and references. For example, BERTScore (Devlin et al., 2018) uses a BERT model to generate token embeddings, and then calculates precision, recall and F1 score based on the dot product between output token embedding and reference token embedding. MoverScore calculates the distance between output and reference embeddings (Zhao et al., 2019).

Supervised metrics A good supervised model should have a high alignment with the human judgement test set. Using statistical features, Stanojević and Sima'an (2014) combines simple features in a linear model and tunes it with human judgements. On the embedding side, the BLEURT (Papineni et al., 2002) model uses a BERT model to encode the output and reference sequences, and provides a similarity score based on a prediction of human judgement based on vector representations. Rei et al. (2020) uses the XLM-RoBERTa (Lample and Conneau, 2019) encoder with pooling layers to tune with a human ranking.

5.3 Other evaluations

Proxy metrics Proxy metrics compare specific aspects of the text such as entity and relation coverage (Goodrich et al., 2019) and text length distribution (Yue et al., 2022) to reflect the text similarity. These metrics only focus on specific properties of the generated texts.

Corpus Level metrics Aggregated metrics at the corpus level are widely used in OGT due to the challenge of obtaining human references. Statistics-based measures compare the model distribution with human distribution based on corpus statistics, such as the amount of repetition (Holtzman et al., 2019), the diversity of n-grams in the generated text (Self-BLEU) (Zhu et al., 2018), generation perplexity to measure how well the generated texts align with human language patterns (Fan et al., 2018), and distribution divergence (Pillutla et al., 2021), which measures the KL divergence between human language distribution and model language patterns. These metrics can give a score to the model that generated such a corpus, but cannot give a quality score for each document.

This work ARGENT is, as far as we are aware, unique in the literature. Rather than find a reference for a given text, we pre-train a model on a dataset constructed from pairs of model output proxy and their most similar references, and their similarity scores. The model learns the mapping from output proxy directly to the similarity score without seeing the underlying reference. During application, ARGENT transfers this ability to an unseen text generation model output text, and assigns a score that reflects the quality of the generated text.

6 Conclusion

In this work, we have proposed a unified theory for machine generated text evaluation, that works both for RGT and OGT. We pointed out the lack of focus on gold-standard reference selection and have suggested an output-oriented reference annotation method for OGTs based on existing RGT output correction methods. We have developed ARGENT, a novel auto-evaluation method on OGT language quality evaluation that requires no human annotation. We have used this auto-evaluation model on different text types and compared it to other commonly used methods. These experiments show that ARGENT out performs all other methods with the exception of Mauve with web text, to which it is ranked second. In comparison to Mauve, however, ARGENT does not require a human corpus, and is able to provide a score for individual texts, rather than for the model generating those texts. Finally, we reviewed previous works along axes of reference selection and the use of similarity metrics.

7 Limitations

This paper provides a text corruption pre-training framework as a proxy for synthetic text, but only explores the use of inflection and local shuffling as corruption methods. If corruption methods can be targeted at specific task evaluation criteria and at the mistakes actually made in synthetic texts, auto-evaluation model could be improved.

The experiments in this work only focus on language quality of texts. More advanced generative models have less language problems, but face other problems such as machine-like responses and hallucination. Expansion of corruption retraining towards these issues could be of interest.

We have not carried out experiments on output-oriented human annotation due to the time and labour costs. Work on output-oriented references using up-to-date similarity metrics and covering a broader range of datasets is expected to further support this theory.

8 Ethical Considerations

As this is a work on the evaluation of generated text quality, rather than the generation of text itself, it has minimal ethical impact. The possible impacts of this work are

- We have provided a new evaluation paradigm with which researchers can work.
- The ARGENT evaluation model provides a measure of the language quality of generated text, thus enabling better decisions on which generated texts to use for a given use case.
- ARGENT only considers language quality, and not the content of generated text. In any text generation task, content should also be considered.

The use of clinical reports was approved by (redacted for anonymisation), with facility for patient opt-out. The reports were stored and processed in an approved, secure environment by authorised researchers. We do not report any individual data from the reports.

The use of Mauve annotated data (Pillutla et al., 2021) and synthetic academic data (Liyanage et al., 2022) are under GNU licence 2.0. BLEU (Papineni et al., 2002) code is under BSD 3-Clause. ROUGE (Lin, 2004) and BLEURT (Sellam et al., 2020) code are under Apache 2.0. BERTScore (Zhang et al., 2019) code is under MIT. All with intended use.

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.

Manik Bhandari, Pranav Gour, Atabak Ashfaq, Pengfei Liu, and Graham Neubig. 2020. Re-evaluating evaluation in text summarization. *arXiv preprint arXiv:2010.07100*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Elizabeth Clark, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah A Smith. 2021. All that's' human'is not gold: Evaluating human evaluation of generated text. *arXiv preprint arXiv:2107.00061*.

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*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*.

Ben Goodrich, Vinay Rao, Peter J Liu, and Mohammad Saleh. 2019. Assessing the factual accuracy of generated text. In *proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 166–175.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Marzena Karpinska, Nader Akoury, and Mohit Iyyer. 2021. The perils of using mechanical turk to evaluate open-ended text generation. *arXiv preprint arXiv:2109.06835*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Vijini Liyanage, Davide Buscaldi, and Adeline Nazarenko. 2022. A benchmark corpus for the detection of automatically generated text in academic publications. *arXiv preprint arXiv:2202.02013*.

Renqian Luo, Liai Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, and Tie-Yan Liu. 2022. Biogpt: generative pre-trained transformer for biomedical text generation and mining. *Briefings in bioinformatics*, 23(6):bbac409.

John I Marden. 1996. *Analyzing and modeling rank data*. CRC Press.

Antonis Maronikolakis, Hinrich Schutze, and Mark Stevenson. 2020. Identifying automatically generated headlines using transformers. *arXiv preprint arXiv:2009.13375*.

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.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. Mauve: Measuring the gap between neural text and human text using divergence frontiers. *Advances in Neural Information Processing Systems*, 34:4816–4828.

Mark A Przybocki, Gregory A Sanders, and Audrey N Le. 2006. Edit distance: A metric for machine translation evaluation. In *LREC*, pages 2038–2043.

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

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

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231.

Miloš Stanojević and Khalil Sima'an. 2014. Beer: Better evaluation as ranking. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 414–419.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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

Xiang Yue, Huseyin A Inan, Xuechen Li, Girish Kumar, Julia McAnallen, Hoda Shajari, Huan Sun, David Levitan, and Robert Sim. 2022. Synthetic text generation with differential privacy: A simple and practical recipe. *arXiv preprint arXiv:2210.14348*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

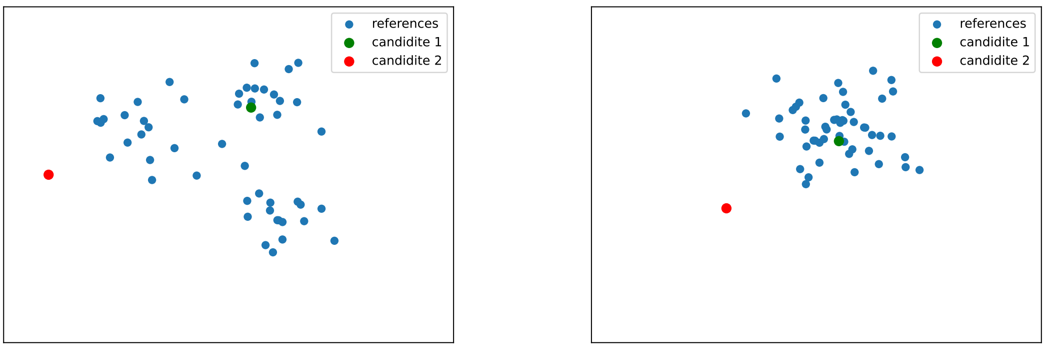
Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M Meyer, and Steffen Eger. 2019. Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance. *arXiv preprint arXiv:1909.02622*.

Yongxin Zhou, Fabien Ringeval, and François Portet. 2023. A survey of evaluation methods of generated medical textual reports. In *Proceedings of the 5th Clinical Natural Language Processing Workshop*, pages 447–459.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 1097–1100.

A Effects of references and similarity functions

The illustrative graph 3 visualises the effects of references and similarity functions. The graph shows a toy 2-D version of space where the Euclidean distance between two points in this graph represents the the similarity score between them defined by some similarity function. In each space, blue dots represent all the gold-standard references, and two candidates of machine output are marked by green and red. In this graph, we can see that the red point is an worse candidate compare to red. But if we chose the left most reference, then the red point would have a higher score. For example, this can the case in our example where "He truly is a clever dog" translation scores higher with certain references. But according to our evaluation theory, the score of the green candidate should be defined by the blue dot closes to it which is the one right on top of it, and the score of the red candidate is defined by the closest blue dot on its right. This will give us a correct judgement that the green candidate is a better candidate than the red one. 3(b) shows a space using better similarity function for example, BERT score versus BLEU. we can see that this similarity function has better ability to cluster the acceptable references closer than 3(a), This reduces the variability in the scores due to different reference choices. In this graph, if we chose the reference on the left, the distance to the red dot is not so close compared to that to the green one. But this may not solve the problem. The selection of closest reference is still not replaceable in most tasks especially larger the reference space.



(a) some similarity function space

(b) a better similarity function space

Figure 3: Illustration of effects of reference points and similarity function

B Text Corruption Methods

Algorithm 1 Token Inflection

```
Define pos_list, inflection_probability, initialise inflected_text ← empty string ""
for current_token in text do
    if draw from inflection_probability then
        current_pos ← pos_tagger(sentence, current_token)
        inflected_pos ← pos_list - current_pos
        inflected_token ← inflection(token, inflected_pos)
        inflected_text ← inflected_text+" "+inflected_token
    end if
end for
return inflected_text
```

Algorithm 2 Token shuffling

```
Define window_range, shuffling_probability, initialise shuffled_text ← empty string "", remain_text ←
text
while len(remain_text)>0 do
    if draw from shuffling_probability then
        draw win_length from window_range
        curr_text←remain_text[:win_length]
        shuffled_text ← shuffled_text +" "+ shuffle(current_text)
        remain_text ← remain_text-curr_text
    end if
end while
return shuffled_text
```

Algorithm 3 Text Corruption with corruption count based score

```
Define corruption method set K, prob range p_range, initialise corr_data
for text n in N do
    initialise corr_count = 0
    for corruption method k in K do
        prob ← random(0, prob_range)
        corr_text = corr_method_k(text, prob)
        for i in text length do
            if corr_text[i] != text[i] then
                corr_count ← corr_count + 1
            end if
        end for
    end for
    score = 1-corr_count/len(K)*N
    corr_data append (corr_text, score)
end for
return corr_data
```

C Hyper-parameter tuning for WebText evaluation

Score	Prob	Inflection			Shuffling		
		Human-like	Sensible	Interesting	Human-like	Sensible	Interesting
Count	0-0.2	83.3	71.4	69.0	0-0.2	85.7	81.0
	0-0.4	83.3	71.4	69.0	78.6	76.2	61.9
	0-0.6	69.0	57.1	45.2	81.0	73.8	66.7
	0-0.8	83.3	76.2	69.0	85.7	81.0	73.8
	0-1.0	66.7	52.4	54.8	81.0	78.6	66.7
BLEURT	0-0.2	-47.6	-52.4	-61.9	-40.0	-45.0	-51.7
	0-0.4	47.6	35.7	35.7	-59.5	-64.3	-81.0
	0-0.6	64.3	54.8	52.4	-9.52	-14.3	-40.5
	0-0.8	81.0	73.8	66.7	-90.5	-90.5	-97.6
	0-1.0	81.0	73.8	66.7	-38.1	-40.0	-57.1
Shufflection (Prob: Shuffling, Inflection)							
Count	0-0.2, 0-0.4	88.1	78.6	76.2	86.7	80.0 3	76.7
	0-0.2, 0-0.8	88.1	78.6	76.2	70	61.7	60
	0-0.8, 0-0.4	88.1	78.6	76.2	79.9	71.7	66.7
	0-0.8, 0-0.8	85.7	76.2	71.4	78.36	70.0	63.3

Table 5: Hyper-parameter tuning: inflection on webtext data

Table 5 shows no great differences between shuffling and inflection. Interestingly, a BLEURT-based score does not give a high score in most cases

D Hyper-parameter Tuning for Synthetic Academic Publications

method	score	0-0.2	0-0.4	0-0.6	0-0.8	0-1.0
Inflection	Count	58	52	59	51	52
	BLEURT	85	79	97	86	80
Shuffling	Count	69	69	68	67	63
	BLEURT	93	77	64	91	75

Table 6: Hyper-parameter tuning: synthetic academic publications

From the Table 6, we can see that the model using BLEURT-based score tends to be the best for this task, and the difference of using inflection or shuffling method is not very significant.

E Hyper-parameter tuning for clinical text evaluation

The clinical reports include five types: Colonoscopy, Gastroscopy, Endoscopic ultrasound (EUS), Sigmodoiscopy and Endoscopic Retrograde Cholangiopancreatography (ERCP). The number of training and testing samples for each type can be found in Table 7. Table 8 shows that with count-based score models, the performance for colonoscopy, gastroscopy and flexible sigmoidoscopy tends to be better than the performance of EUS and ERPC.

Model	Prob	Col	Endo	ERCP	Gstr	Sig	Total
train	20411	2009	1348	40658	9453	243	74122
valid	3676	971	784	10263	2790	46	18530
total	24087	2980	2132	50948	12243	289	92652

Table 7: Statistics of clinical data

Score	Prob	Col	Endo	ERCP	Gstr	Sig	Total
Inflection							
Count	0-0.2	66.1±7.9	60.5±10.6	58.0±9.9	67.9±11.2	67.5±13.8	64.0±4.7
	0-0.4	70.1±6.6	62.9±10.5	64.6±12.7	70.9±9.3	71.8±10.9	**68.1±2.4**
	0-0.6	66.9±6.1	56.0±11.3	61.8±10.4	66.9±11.0	72.1±10.6	64.7±4.2
	0-0.8	68.8±8.8	62.4±11.1	61.7±10.1	70.6±8.3	71.0±9.3	66.9±2.9
	0-1.0	69.6±5.6	59.6±13.0	62.9±9.3	72.6±10.2	70.7±9.0	67.1±3.1
BLEURT	0-0.2	58.1±12.1	56.1±9.8	56.2±9.2	61.3±15.6	54.8±11.0	57.3±6.3
	0-0.4	59.1±12.3	55.5±10.0	54.2±10.0	60.1±16.0	54.8±11.0	56.7±6.1
	0-0.6	59.3±12.3	54.8±9.2	54.5±9.3	60.4±15.0	57.0±11.4	57.2±5.8
	0-0.8	60.4±12.3	56.5±10.2	56.1±8.9	60.4±15.3	56.7±10.9	58.0±6.4
	0-1.0	60.5±11.1	56.4±9.4	58.5±9.2	60.9±14.9	57.0±10.4	**58.7±5.8**
Shuffling							
Count	0-0.2	66.1±8.5	63.7±11.3	62.2±10.7	69.7±13.9	67.7±12.9	65.9±3.8
	0-0.4	82.9±8.2	76.3±8.0	74.0±7.6	81.6±9.8	81.7±12.0	**79.3±2.6**
	0-0.6	74.6±5.7	60.9±10.7	67.4±8.4	73.9±12.1	73.5±10.2	70.0±2.6
	0-0.8	64.9±7.8	58.4±8.5	61.2±10.1	65.4±13.8	60.5±12.5	62.1±2.6
	0-1.0	71.6±8.4	66.7±10.6	67.9±10.2	75.1±13.0	68.4±13.5	69.9±3.4
BLEURT	0-0.2	54.8±14.5	55.4±9.5	58.7±8.1	59.0±15.6	53.1±10.4	56.2±6.2
	0-0.6	54.2±14.1	55.7±9.4	58.8±8.6	58.6±15.6	53.9±10.5	56.2±6.2
	0-0.6	54.5±14.5	55.8±10.6	59.7±6.7	58.2±15.5	53.6±10.2	56.3±6.4
	0-0.8	55.7±13.1	54.8±10.2	59.2±8.1	59.5±16.1	53.7±9.6	56.6±6.0
	0-1.0	54.4±13.7	55.3±10.4	59.8±8.3	59.6±15.1	55.0±10.0	**56.8±6.4**
Shufflection (Prob: Shuffling, Inflection)							
Count	0-0.4, 0-0.4	64.6±7.4	60.2±7.4	62.1±10.0	67.1±15.4	64.8±11.4	63.8±3.2
	0-0.4, 0-1.0	66.6±7.6	57.4±8.3	62.1±11.1	68.2±12.6	63.4±11.4	63.9±3.1
	0-0.6, 0-0.4	66.3±6.8	59.8±9.0	60.9±9.3	66.6±13.4	64.6±10.4	63.6±3.3
	0-0.6, 0-1.0	80.6±8.1	57.2±6.2	64.3±11.1	69.1±13.6	67.3±11.7	**67.7±3.5**
BLEURT	0-1.0, 0-1.0	58.3±11.8	56.4±10.5	59.5±74.1	59.6±16.2	57.4±10.5	58.2±6.4
	0-1.0, 0-0.8	60.4±13.5	55.8±11.7	59.7±8.5	62.1±15.3	58.6±9.7	59.3±6.3
	0-0.8, 0-1.0	60.5±12.2	57.1±9.9	59.2±9.0	62.0±14.2	58.1±9.9	**59.4±6.1**
	0-0.8, 0-0.8	60.7±11.9	55.4±9.7	59.3±8.7	61.0±16.2	57.5±9.9	58.8±5.6

Table 8: Hyper-parameter tuning on clinical reports