CLEval: Character-Level Evaluation for Text Detection and Recognition Tasks

Youngmin Baek[1], Daehyun Nam[1], Sungrae Park[1], Junyeop Lee[1],
Seung Shin[1], Jeonghun Baek[1], Chae Young Lee[2] and Hwalsuk Lee*[1]

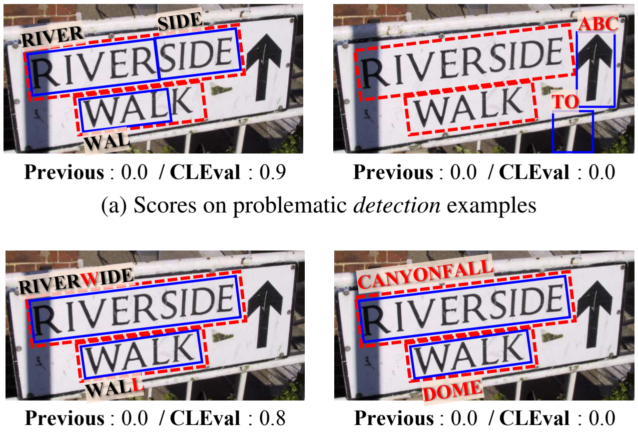
[1]Clova AI Research, NAVER Corp.
[2]Yale University

Abstract

Despite the recent success of text detection and recognition methods, existing evaluation metrics fail to provide a fair and reliable comparison among those methods. In addition, there exists no end-to-end evaluation metric that takes characteristics of OCR tasks into account. Previous end-to-end metric contains cascaded errors from the binary scoring process applied in both detection and recognition tasks. Ignoring partially correct results raises a gap between quantitative and qualitative analysis, and prevents fine-grained assessment. Based on the fact that character is a key element of text, we hereby propose a Character-Level Evaluation metric (CLEval). In CLEval, the instance matching process handles split and merge detection cases, and the scoring process conducts character-level evaluation. By aggregating character-level scores, the CLEval metric provides a fine-grained evaluation of end-to-end results composed of the detection and recognition as well as individual evaluations for each module from the end-performance perspective. We believe that our metrics can play a key role in developing and analyzing state-of-the-art text detection and recognition methods. The evaluation code is publicly available at `https://github.com/clovaai/CLEval`.

1. Introduction

Along with the progress in the field of machine learning, the performances of text detectors and recognizers have remarkably improved over the past few years [15, 17, 14, 2, 1, 11]. However, existing detection and recognition evaluation metrics fail to provide a fair and reliable comparison among those methods. Especially when evaluating end-to-end models, errors are aggregated due to unconvincing measurements in each part. Fig. 1 illustrates common problems encountered in end-to-end detection and recognition modules. The previously used instance-level binary scoring process assigns a value of 0 on both decent(left figures) and wrong(right figures) results.


RIVER
SIDE
WAL
ABC
TO
Previous : 0.0 / CLEval : 0.9
Previous : 0.0 / CLEval : 0.0
RIVERWIDE
WALL
CANYONFALL
DOME
Previous : 0.0 / CLEval : 0.8
Previous : 0.0 / CLEval : 0.0

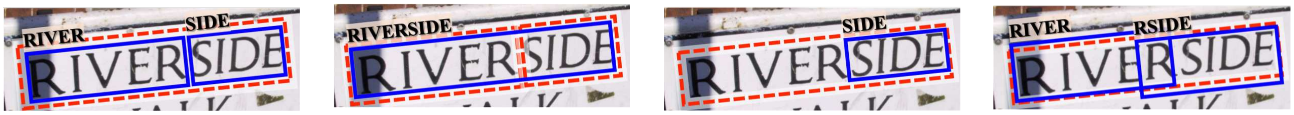
(a) Scores on problematic *detection* examples

(b) Scores on problematic *recognition* examples

Figure 1: Comparison of CLEval with previous end-to-end metric[6]. Figures in the left column are slightly wrong cases while figures in the right are completely wrong cases. Red: GT. Blue: detection. Texts indicate the recognized results of the detection box, and texts in red are incorrect.

To better understand where established text detection and recognition evaluation metrics fail, a closer look into the intrinsic nature of texts is required. At a fundamental level, text consists of words, which can further be decomposed into an array of characters. This character array embodies two intrinsic characteristics: its sequential nature and its content. Text detection's goal of locating words can then be reinterpreted as finding the area that encapsulates the right sequence and content in a group of characters. The degree to which the right sequence and content are recognized within the detection area denotes the *granularity* and *correctness* issues, respectively. A more specific explanation of these attributes follows – with examples of what they measure in Figure 2.

*Corresponding author.



(a) Split case (b) Merged case (c) Missing characters (d) Overlapping characters

Figure 2: Examples of issues for the fair evaluation of text detection and recognition tasks. (a) and (b) correspond to the issue of granularity, (c) and (d) are related to the issue of correctness.

Granularity is the degree to which the text detection model captures the sequence of characters in exactly one word as one unbroken sequence. Split detection results (Figure 2(a)) break the character sequence in the word and merged detection results (Figure 2(b)) fail to capture exactly one word. Both cases incur penalties proportionate to the number of splits or merges per word.

Correctness is the degree to which the text detection and recognition model captures the content of word. Specifically each character in that word must be detected and recognized exactly once with its right order. A penalty is incurred proportionate to the number of missing or overlapping characters in both detection and recognition results (Figure 2(c, d)).

Majority of the public datasets provide a word-level annotation since each word contains a semantic meaning. However, as Fig. 1 shows, evaluating word-level boxes with a predefined threshold provokes various issues. Despite having appropriate box predictions, the binary scoring process discards acceptable prediction results and produces unexplainable scores. To provide more detailed interpretation of the models, recent studies have adopted a character-level evaluation process [8, 9]. Inspired by them, our proposed metric, named as CLEval (Character-Level Evaluation), is designed to perform end-to-end evaluations without explicit character annotations. The method adopts two key components; instance matching process and character scoring process. The instance matching process solves granularity issues by pairing all possible GT and detection boxes that share at least one character, and the character-level scoring process solves correctness issues by calculating the longest common subsequence between GT and predicted transcriptions.

The CLEval metric is primarily designed to evaluate end-to-end tasks, but it can also be applied to individual detection and recognition modules. Interpretation of each module is valuable since it allows us to discover how each component affects overall performance. Assuming that the characters are evenly placed within a word box, the detection evaluation is conducted using pseudo-character center positions. This method was first proposed by [8], but we further developed the idea to handle end-to-end models based on the same scoring policy. The proposed metric also provides quantitative indicators of recognition modules by measuring correctly recognized words within detection boxes.

The main contributions of this paper can be summarized as follows. 1) We propose a character-level evaluation metric that is favorable qualitative view without character-level annotations. 2) We define granularity and correctness issues, and solve them by performing instance matching and character scoring processes. 3) We propose a unified protocol that could evaluate not only end-to-end tasks, but also individual detection and recognition modules.

2. Related works

2.1. Detection evaluation

Intersection-over-Union (IoU) IoU metric originally comes from object detection task such as Pascal VOC [5]. IoU accepts detections that match the ground truth (GT) box in an exclusive one-to-one manner only when the overlapping region satisfy the predefined threshold. Although IoU is the most widely used evaluation metric thanks to its simplicity, its behavior is clearly not suitable for evaluating texts as argued by [3, 13]. IoU cannot handle granularity and correctness issues, which is critical for OCR tasks.

DetEval DetEval [16] was designed to solve the granularity issue by allowing multiple relationships of a single bounding box. Their matching processes are conducted by accepting one-to-one, one-to-many, and many-to-one relationships. However, each instance is evaluated based on both area recall and area precision thresholds. Area-based threshold not only causes correctness issues, but also has a limitation when trying to apply end-to-end evaluation.

Tightness-aware IoU (TIoU) Liu et al. recently suggested the TIoU metric that penalizes based on the occupation ratio between detection and ground truth. By doing that, TIoU tried to give the high score to more similar detection compared to the box of ground truth. The major weakness is that TIoU penalizes slight differences between the ground truth and detection, even if the recognition results of those detection boxes are same. This is far from the end user perspective, and it is unfair if the correct box got a different score under the TIoU metric due to the small perturbation of box size.

TedEval A character-level evaluation metric for text detection has been proposed by [8]. The metric alleviates qualitative disagreements, but can only be used to evaluate text detection modules. We adopt the idea of using pseudo-

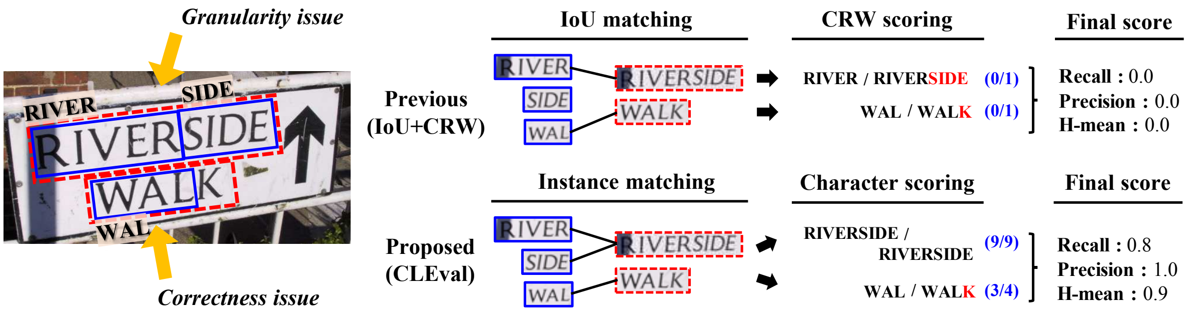


Figure 3: Overall evaluation process of IoU+CRW and CLEval metrics. In IoU+CRW, only two bounding boxes are matched through 0.5 IoU threshold, and partially detected texts are discarded in CRW scoring process. In CLEval, all detection boxes are matched, and correct numbers of characters are reflected to the final evaluation.

characters, and apply a character-level evaluation process to evaluate end-to-end results.

2.2. Recognition evaluation

Correctly Recognized Words (CRW) As its term suggests, the CRW is a binary score metric by judging correct answers when the transcription and the recognition result are exactly same. This method has a fundamental limit of a binary score system that fails to give different scores to an absurd recognition result and an almost accurate result.

Edit Distance [10] The Edit Distance (ED) method is a common algorithm used to quantify how dissimilar two given strings are. The ED of two strings is the minimum operation required to transform one string into another. In the most standard Levenshtein distance calculation, such an operation involves insertion, deletion, and substitution. Utilizing ED to evaluate scene text recognition model is considered reasonable in that the score reflects how well the model is performing with distance measures. Longest Common Subsequence (LCS) is literally the longest common subsequence in a set of sequences, and is a specialized case of ED which only uses insertion and deletion operations [12].

2.3. End-to-end evaluation

IoU and CRW The IoU and CRW are strictly a cascaded evaluation metric. The detection stage filters out detection results whose IoU with the corresponding GT is below the threshold. Matches with IoU are judged by the CRW. Both metrics in each of the stages are reported to have hindrances for fine-grained assessment due to the binary chain scoring.

PopEval PopEval was proposed to make full use of text information from recognition results. Their character elimination process is simple yet good enough from a practical point of view. However, PopEval does not provide detection evaluation. We adopted the idea of character elimination, and enhanced by using a substring elimination scheme to mitigate the problem of ignoring the order of texts.

3. Methodology

Fig. 3 shows comparison of our method with the IoU + CRW metric. Detection boxes in the word “RIVERSIDE” show the granularity issue, and boxes in the word “WALK” show the correctness issue. While previous metrics fail to accept decent prediction results, our metric successfully quantifies various conditions through the matching process and the scoring process. The matching process identifies instance-level pairs between GT and detected boxes, and the scoring process provides a final score by analyzing extracted statistics.

3.1. Matching process

In this section, the matching process is explained in detail. First, to overcome the absence of character annotations, we adopt the idea from [8] and calculate Pseudo-Character Center(PCC) positions. GT and detection boxes are considered a match if they satisfy two conditions. Their overlapping regions between GT and detection must share at least one PCC in common and should also cover an adequate GT area. Any candidates that do not satisfy the conditions are filtered out from the matching process.

3.1.1 Pseudo-Character Center (PCC)

We first need to know the location of the characters to identify whether a character region is covered by a detection box. However, most of the public datasets only provide word-level bounding box annotations. To handle this issue, we synthetically generate Pseudo-Character Center(PCC) points using GT word box and transcription. Let $\boldsymbol{G} = \{G_1, ..., G_I\}$ be a set of GT boxes and $\boldsymbol{D} = \{D_1, ..., D_J\}$ be a set of detected boxes where I and J denote the size of

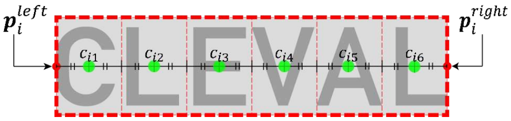


Figure 4: An example of computing PCC of G_i. Green dot: PCC. Red dash: pseudo character box. Grey: G_i.

each set. Each GT box, G_i, contains a word with l_i^G characters. As shown in Fig. 4, we compute the k-th PCC of G_i to obtain the positional information of the characters,

$$c_i^k = \left(\frac{2k-1}{2l_i^G}\right) p_i^{\text{left}} + \left(1 - \frac{2k-1}{2l_i^G}\right) p_i^{\text{right}} \tag{1}$$

where p_i^{left} and p_i^{right} indicate midpoints located on the left and right edges of G_i. The equation allows us to construct PCC points using both quadrilateral and polygon boxes. PCC point generation for polygon boxes is described in the Appendix.

The PCC points are generated under the assumption that the characters are evenly divided within a word box. However, constructed points may not be perfectly aligned with actual character positions because characters of different sizes coexist in the word image. Even with this ambiguity, our assumption works fairly well in most cases. Fig. 5 shows constructed PCC points on real datasets.

3.1.2 Matching based on character inclusion

The first criterion for finding a match between GT and detection box is the inclusion of at least one PCC point. Here, we define character inclusion candidate, $\hat{m}_{ij}^k$, between D_j and G_i as

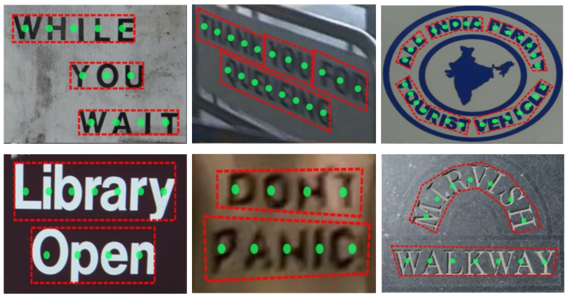
$$\hat{m}_{ij}^k = \mathbb{I}(c_i^k \text{ in } \mathrm{D}_j) \tag{2}$$

where $\mathbb{I}(\mathrm{A})$ is a conditional function that gives a value of 1 when A is satisfied and 0 otherwise. If $\hat{m}_{ij}^k = 1$, it is probable that G_i and D_j is matched since they share at least one PCC in common.

The second matching criterion is the area of intersection between detection box and GT text region. Since PCC is a single coordinate in the image, inclusion of a point does not guarantee good localization of the ground truth text region. In order to alleviate this ambiguity, detection boxes that cover small GT box regions are filtered out. To this end, the area precision of D_j is defined as follow

$$AreaPrecision_j = \frac{Area(\cup_{i \in \{i \| \exists k,\ \hat{m}_{ij}^k = 1\}} (\mathrm{D}_j \cap \mathrm{G}_i)))}{Area(\mathrm{D}_j)}, \tag{3}$$

where the union condition, $\{i \| \exists k,\ \hat{m}_{ij}^k = 1\}$, indicates a set of GT boxes that contains at least one of its PCC points matched with the D_j. The matching process filters out candidates whose non-text region is larger than the text region.



(a) ICDAR2013[7] (b) ICDAR2015[6] (c) TotalText[4]

Figure 5: Visualization of PCC points on different ground-truth annotations.

Therefore, the final box matching flags M_{ij} are defined by considering the character inclusion flag m_{ij}^k and its area precision as

$$\begin{aligned} m_{ij}^k &= \hat{m}_{ij}^k \times \mathbb{I}(AreaPrecision_j > 0.5), \\ M_{ij} &= \mathbb{I}(\sum_k m_{ij}^k > 0). \end{aligned} \tag{4}$$

To solve the granularity issue, we need to consider one-to-many and many-to-one cases. In our matching process, $AreaPrecision_j$ explicitly handles one-to-one and many-to-one cases by calculating the union of intersections between matched G_is and D_j. In our metric, one-to-many match does not need to be processed since each split detection box is matched with one GT box by checking the inclusion of at least one character.

3.1.3 Summarized matching statistics

Table. 1 shows matching flags and statistics. The upper and lower cases mean box-level and the character-level instances, respectively. The subscript indicates the box index, and the superscript represents the character index. The statistics are written in the script font, and they represent row wise and column wise summations.

		D_1		...	D_j		Stat.		Recall
G_1	c_1^1 c_1^2 ... c_1^k	M_{11}	m_{11}^1 m_{11}^2 ... m_{11}^k		M_{1j}	m_{1j}^1 m_{1j}^2 ... m_{1j}^k	$\mathscr{G}_1$	$\mathscr{g}_1^1$ $\mathscr{g}_1^2$ ... $\mathscr{g}_1^k$	R_1
...	...	...	...		...	...	...	...	...
G_i	c_i^1 c_i^2 ... c_1^k	M_{i1}	m_{i1}^1 m_{i1}^2 ... m_{i1}^k		M_{ij}	m_{ij}^1 m_{ij}^2 ... m_{ij}^k	$\mathscr{G}_i$	$\mathscr{g}_i^1$ $\mathscr{g}_i^2$ ... $\mathscr{g}_i^k$	R_i
Stat.		$\mathscr{D}_1$	$\mathscr{d}_1$		$\mathscr{D}_j$	$\mathscr{d}_j$			
Precision		P_1		...	P_j				

Table 1: Table of box-level and character-level matching flags and their statistics.

Matching statistics are summarized in Table 2. The values are obtained using the character inclusion flag m_{ij}^k and the box matching flag M_{ij}. By aggregating matching statistics, we could identify total number of box matches and character inclusions. The value of $\mathcal{G}_i$ and $\mathcal{D}_j$ show the number of matched box candidates on each G$_i$ and D$_j$. The number $\mathscr{g}_i^k$ denotes matched detection boxes on k-th PCC point of G_i, and $\mathscr{d}_j$ shows the number of PCC points covered by a detection box D_j. These statistics are finally used to calculate character scores and granularity penalties.

Stat.	Eq.	Description
$\mathcal{G}_i$	$\sum_j M_{ij}$	number of $\boldsymbol{D}$ matched with G$_i$
$\mathcal{D}_j$	$\sum_i M_{ij}$	number of $\boldsymbol{G}$ matched with D$_j$
$\mathscr{g}_i^k$	$\sum_j m_{ij}^k$	number of $\boldsymbol{D}$ including k-th character of G$_i$
$\mathscr{d}_j$	$\sum_{ik} m_{ij}^k$	number of characters in G$_i$s matched with D$_j$

Table 2: Description of matching statistics.

3.2. Scoring process

Once the matching candidates are obtained, we now evaluate character-level correctness. Eq. 5 is the ground rule to calculate recall and precision.

$$Score = \frac{CorrectNum - GranulPenalty}{TotalNum} \tag{5}$$

TotalNum represents the number of GT or detected characters, and *CorrectNum* denotes the number of correct characters. *GranulPenalty* is proportional to the split number of GT or detection boxes. Each attribute will be explained in the following subsections.

3.2.1 *TotalNum*: Total Number of Characters

TotalNum, the denominator in Eq. 5, indicates the number of target characters. When evaluating the recall of G$_i$, $TotalNum_i^G$ is set to the GT text length, l_i^G. However, note that the value of text length is absent when measuring detector accuracy. The value of $TotalNum_j^D$ differs depending on the availability of word transcriptions.

For end-to-end evaluation, l_j^D, which is the length of the predicted text in D$_j$, can be used to represent $TotalNum_j^D$. However, for detection evaluation, the length of predicted word transcription is unknown. In this case, we define $TotalNum_j^D$ using $\mathscr{d}_j$ in Table. 2, where it defines the number of included PCC points of all the matched GT boxes.

3.2.2 *CorrectNum*: Correct Number of Characters

Correct Number for end-to-end evaluation Since word transcriptions are available when performing an end-to-end evaluation, the number of correct characters can be measured by finding a subsequence between the transcriptions of matched GTs and detection boxes. However, multiple matches could occur during the instance matching process, and thus, a character score could be calculated multiple times. To avoid this problem, we introduce *Subsequence Elimination Scoring Process (SESP)* that calculates each character score once and eliminates the matched subsequences in both GTs and predictions.

SESP is described in Algorithm 1. For each GT box, a set of matched detection boxes are collected and sorted according to the order of included PCC points. Given sorted detection boxes, word transcriptions are assembled together to form a single word(*recog_text*). We then extract *common_seq*, which is the Longest Common Sequence(LCS)[12] between GT and *recog_text*. The length of *common_seq* is directly used as $CorrectNum_i^G$. For each matched detection box, *det_seq* is extracted between the *common_seq* and D$_j$, and the length of *det_seq* within all matched GTs is accumulated to $CorrectNum_j^D$. Finally, the *det_seq* gets eliminated in both D$_j^{text}$ and *common_seq*. This elimination process is required to avoid multiple matches between detection and GT transcriptions. Figure 6 shows an example of SESP on split and merge cases.

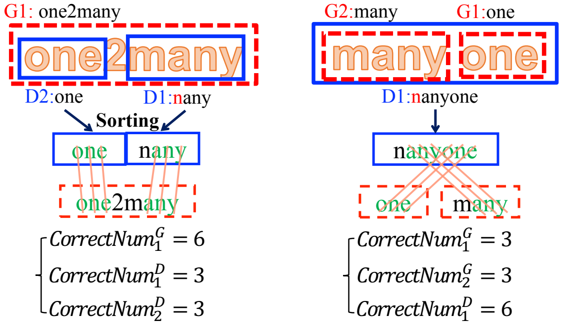


Figure 6: Examples of SESP on different matched cases: split case on left, merge case on right.

Note that unlike the PopEval[9], which ignores the order of characters for scoring, we perform SESP after ordering detection boxes into the right sequence. Therefore, when performing evaluation, the order of the characters are taken into account. Our metric is almost free from having errors due to character permutations.

Correct Number for detection evaluation

Word transcription is not available when evaluating detection results. Therefore, we utilize the number of PCC inclusion since the detection accuracy is related to whether a detected box covers a character or not. The number of correct characters, *CorrectNum*, is defined using detection

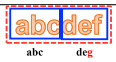
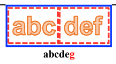
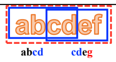
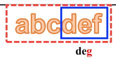
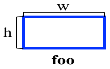
Evaluation			Detection				End-to-end			
Score			Recall		Precision		Recall		Precision	
Granularity	Split		$\frac{6-1}{6}$	$\frac{\mathbf{5}}{\mathbf{6}}$	$\frac{3-0}{3} \oplus \frac{3-0}{3}$	$\frac{\mathbf{6}}{\mathbf{6}}$	$\frac{5-1}{6}$	$\frac{\mathbf{4}}{\mathbf{6}}$	$\frac{3-0}{3} \oplus \frac{2-0}{3}$	$\frac{\mathbf{5}}{\mathbf{6}}$
	Merge		$\frac{3-0}{3} \oplus \frac{3-0}{3}$	$\frac{\mathbf{6}}{\mathbf{6}}$	$\frac{6-1}{6}$	$\frac{\mathbf{5}}{\mathbf{6}}$	$\frac{3-0}{3} \oplus \frac{2-0}{3}$	$\frac{\mathbf{5}}{\mathbf{6}}$	$\frac{5-1}{6}$	$\frac{\mathbf{4}}{\mathbf{6}}$
Correctness	Overlapping		$\frac{6-1}{6}$	$\frac{\mathbf{5}}{\mathbf{6}}$	$\frac{3-0}{4} \oplus \frac{3-0}{4}$	$\frac{\mathbf{6}}{\mathbf{8}}$	$\frac{5-1}{6}$	$\frac{\mathbf{4}}{\mathbf{6}}$	$\frac{4-0}{4} \oplus \frac{1-0}{4}$	$\frac{\mathbf{5}}{\mathbf{8}}$
	Missing		$\frac{3-0}{6}$	$\frac{\mathbf{3}}{\mathbf{6}}$	$\frac{3-0}{3}$	$\frac{\mathbf{3}}{\mathbf{3}}$	$\frac{2-0}{6}$	$\frac{\mathbf{2}}{\mathbf{6}}$	$\frac{2-0}{3}$	$\frac{\mathbf{2}}{\mathbf{3}}$
Etc	FalsePositive		$\frac{0-0}{0}$	$\frac{\mathbf{0}}{\mathbf{0}}$	$\frac{0-0}{round(w/h)}$	$\frac{\mathbf{0}}{\mathbf{3}}$	$\frac{0-0}{0}$	$\frac{\mathbf{0}}{\mathbf{0}}$	$\frac{0-0}{3}$	$\frac{\mathbf{0}}{\mathbf{3}}$
Scoring policy			**Recall** $= \frac{\sum_{i=1}^{\lvert G \rvert} (CorrectNum_i^G - GranulPenalty_i^G)}{\sum_{i=1}^{\lvert G \rvert} TotalNum_i^G}$, **Precision** $= \frac{\sum_{j=1}^{\lvert D \rvert} (CorrectNum_j^D - GranulPenalty_j^D)}{\sum_{j=1}^{\lvert D \rvert} TotalNum_j^D}$							

Table 3: Comprehensive examples of scoring for issues in text evaluations. Note that the scores are expressed as fractions rather than reduction because each denominator has the important meaning; that is character length. ⊕ indicates the separated summation of each nominator and denominator.

Algorithm 1: Subsequence Elimination Scoring Process (SESP)

```
for $G_i$ in $\boldsymbol{G}$
    $\boldsymbol{DG_i} \leftarrow$ a set of the matched $\mathrm{D}_j$s with $\mathrm{G}_i$
    $recog\_text \leftarrow$ a serialized text from $\mathrm{D}_j$ in $\boldsymbol{DG_i}$
                  in the order of the matched PCCs
    $common\_seq \leftarrow$ the Longest Common Subsequence(LCS)
                  between ($\mathrm{G}_i^{\text{text}}$, $recog\_text$ )
    $CorrectNum_i^G \leftarrow \mathrm{len}(common\_seq)$
    for $D_j$ in $\boldsymbol{DG_i}$
        $det\_seq \leftarrow$ the characters from $\mathrm{D}_j$ in $common\_seq$
        $CorrectNum_j^D + = \mathrm{len}(det\_seq)$
        $\mathrm{D}_j^{\text{text}} - = det\_seq$
        $common\_seq - = det\_seq$
    end
end
```

statistics as follows;

$$\begin{aligned} CorrectNum_i^G &= \sum_{k=1}^{l_i} \mathbb{I}\left(g_i^k \geq 1\right), \\ CorrectNum_j^D &= \sum_i \sum_{k=1}^{l_i} \frac{\mathbb{I}\left(g_i^k \geq 1\right)}{\max(g_i^k, 1)}. \end{aligned} \tag{6}$$

$CorrectNum_i^G$ indicates the number of included PCC points of G_i within detection boxes, and $CorrectNum_j^D$ represents the number of accumulated PCC points of D_j within all the matched G_is. Additionally, $CorrectNumj^D$ of each character is divided by the inclusion counts g_i^k to penalize overlapping cases. By doing this, only one of the matched characters is marked correct, and this can be seen from the same perspective of the subsequence elimination process when measuring end-to-end results.

3.2.3 *GranulPenalty*: Granularity Penalty

The granularity indicates the connectivity condition between characters. We define *GranulPenalty* as a penalty representing how much the detection result loses the connectivity information.

From a GT perspective, the most ideal condition is formed when a single detection box is matched($\mathcal{G}_i = 1$). Likewise, from a detection perspective, the most ideal condition is formed when a single GT box is matched($\mathcal{D}_j = 1$). The granularity penalty equation is shown in Eq. 7. As number of $\mathcal{D}_j$ and $\mathcal{G}_i$ grows, penalty increases proportionally.

$$\begin{aligned} GranulPenalty_i^G &= \mathcal{G}_i - 1, \\ GranulPenalty_j^D &= \mathcal{D}_j - 1. \end{aligned} \tag{7}$$

This equations means that the weight of the loss of connectivity is same as the failure to detect a single character.

3.2.4 Character Number of False Positive Detection

An appropriate penalty should be given to false positive(FP) detection, but we can't get the number of characters to penalize FP detection explicitly for detection evaluation. Therefore, the character length of the FP is estimated by assuming that the number of characters is proportional to the aspect ratio to fit into the box. As a result, the *TotalNum* for FP is given through aspect ratio as shown in Eq. 8a. For end-to-end evaluation, the character length of recognized text l_j^D in the detection box is given, so this value is applied to the *TotalNum* of FP as shown in Eq. 8b.

$$TotalNum_{j|\mathcal{D}_j=0}^D = round(w/h) \tag{8a}$$

$$TotalNum_{j|\mathcal{D}_j=0}^D = l_j^D \tag{8b}$$

where h indicates the minimum length of bounding box of the detection, and w indicates the maximum length of it.

3.2.5 Scoring summary

Table 3 shows how scoring is processed on various issues. Basically, instance matching is processed on both detection and end-to-end evaluations. The scoring process, however, differs depending on the level of evaluation. When estimating detection performance, we take the information of inclusive PCC points, and when evaluating end-to-end results, we take the correct subsequence of recognized texts.

In this way, recall and precision of each box instance are obtained. Other evaluation metrics calculate the final score by taking the average of all recall and precision values. This is not the case of our evaluation since the denominator needs to be the sum of character numbers. The final recall and precision values are obtained by separately adding numerator and denominator scores of each instance as

$$\begin{aligned} Recall &= \frac{\sum_{i=1}^{|G|} (CorrectNum_i^G - GranulPenalty_i^G)}{\sum_{i=1}^{|G|} TotalNum_i^G}, \\ Precision &= \frac{\sum_{j=1}^{|D|} (CorrectNum_j^D - GranulPenalty_j^D)}{\sum_{j=1}^{|D|} TotalNum_j^D}. \end{aligned} \tag{9}$$

Finally, *H-Mean* is calculated using Eq. 10 as usual.

$$H\text{-}Mean = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{10}$$

Explicit recognition performance is also important for researchers developing recognition models. Using the attributes introduced in section 3.1, we measure the sole performance of the recognizer.

In order to solely evaluate recognition outputs, it is necessary to eliminate factors coming from detection outputs. One element that does not affect recognition performance is box granularity. We therefore remove granularity penalty when evaluating recognition performance. Additionally, unpaired prediction boxes should also be excluded. End-to-end performance assigns a penalty if a predicted word has no GT pair, and this is not fair since the error propagates from the detection performance. After eliminating the factors that disrupt fair recognition performance, we obtain Eq. 11 that expresses the Recognition Score *(RS)*.

$$RS = \frac{\sum_{j=1}^{|D|} CorrectNum_j^D \times \mathbb{I}(\mathscr{D}_j > 0)}{\sum_{j=1}^{|D|} max(TotalNum_j^D, d_j) \times \mathbb{I}(\mathscr{D}_j > 0)} \tag{11}$$

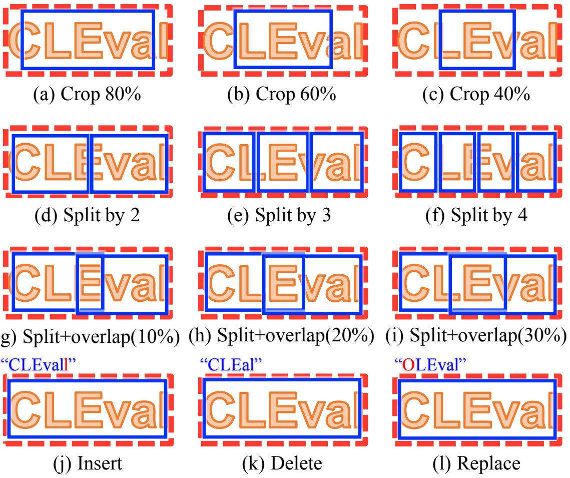
The equation measures recognition performance by dividing the number of correctly recognized characters by the total number of predicted characters matched with the GT instance.

4. Experiments

In this section, for ease of discussion, we analyze the tendency of our metric using the toy-examples constructed on ICDAR2013 dataset. The evaluation of our metric on real detection and recognition outputs are provided in the appendix.

4.1. Toy-example experiments

To compare the characteristics of the evaluation metric, a toyset is designed to reflect the granularity and correctness issues. To evaluate detection performance, nine cases were synthetically generated using ICDAR2013 dataset[7]. The cases are categorized into three parts; crop, split and overlap. To simulate recognition issues, we expect that the synthetic detection results have the same box as the GTs, and modify the text to cover insert, delete, and replace cases. Detailed toy-examples are illustrated in Figure 7.



(a) Crop 80% (b) Crop 60% (c) Crop 40%

(d) Split by 2 (e) Split by 3 (f) Split by 4

(g) Split+overlap(10%) (h) Split+overlap(20%) (i) Split+overlap(30%)

(j) Insert (k) Delete (l) Replace

Figure 7: Detailed cases for toy-example experiment. First row shows crop cases, second row shows split cases, and third row shows overlap cases. Recognition cases such as insert, delete, and replace are in the last row. The number of insertion, deletion and replacement is increased from 1 to 3.

4.2. Detection evaluation

The detection evaluation result of the toy-example experiment is shown in Table 4. DetEval and IoU metrics show typical problems encountered when using a threshold based binary scoring policy. For example, DetEval uses an area recall threshold value of 0.8. Any detection boxes outside this threshold are not considered a match, and therefore, the H-mean values under crop ratio 80% gets a value close to 0. Similar tendency is also found in the IoU metric. The metric uses a threshold value of 0.5, and thus, the H-mean value under crop ratio 50% gets a value close to 0. This indicates that the binary scoring process does not take into account boxes that do not meet predefined threshold conditions.

Case	Detection Metrics								
	DetEval			IoU			CLEval		
	R	P	H	R	P	H	R	P	H
Original	99.7	99.9	99.8	99.8	100	99.9	99.7	98.7	99.2
Crop 80%	97.1	97.3	97.2	99.8	100	99.9	82.7	98.7	90.0
Crop 60%	0.3	0.5	0.4	99.7	99.9	99.8	60.7	98.9	75.2
Crop 40%	0.1	0.1	0.1	0.0	0.0	0.0	40.1	95.2	56.4
Split by 2	78.7	79.9	79.3	98.4	49.3	65.7	82.4	97.2	89.2
Split by 3	78.6	79.8	79.2	0.0	0.0	0.0	66.7	94.2	78.1
Split by 4	78.5	79.7	79.1	0.0	0.0	0.0	53.2	89.9	66.8
Overlap 10%	78.9	79.8	79.4	99.8	50.0	66.6	81.1	88.7	84.7
Overlap 20%	79.0	79.9	79.4	99.8	50.0	66.6	80.9	82.0	81.5
Overlap 30%	79.0	79.8	79.4	99.8	50.0	66.6	80.9	74.0	77.3

Table 4: Comparison of detection evaluation metrics on toy-example from ICDAR2013 dataset. Some scores are highlighted: Red above 95, Blue below 5.

As shown in Table. 4 DetEval and IoU metric produces unreasonable values. DetEval scores in split and overlap cases are almost identical. We expect the scores to be different, but we get the same results because a penalty of 0.8 is assigned in all cases. On the other hand, the IoU recall precision values are strange in split and overlap cases. This is because only one of the detection boxes is paired with the GT box. Recall value of the matched detection box is close to 1 and the precision value of the mismatched detection box is close to 0.5. Also, the DetEval and IoU scores remain the same regardless of the change in overlapping ratio.

While DetEval and IoU metrics fail to cover acceptable detection results, CLEval metric performs fine-grained evaluation on detection results. The calculated recall score in CLEval is proportional to the size of the cropped box region. For the overlapping case, a precision penalty relative to the size of the overlapping region is given. The recall scores in three overlapping cases are almost the same. This is reasonable because every GT character is detected, and the number of detected duplicate characters decrease the precision score.

During the CLEval evaluation process, intermediate attributes directly related to *CorrectNum*, *TotalNum*, and *GranulPenalty* are extracted. These are the number of split and merge, frequency of missing and overlapping characters, and estimated character numbers in false positives. Table 5 shows a summary of extracted intermediate attributes on ICDAR2013 toy dataset. Each quantified attribute conveys a practical view to researchers and end-users. The information can be used by the researchers to further analyze and develop detection models.

4.3. End-to-end evaluation

In end-to-end evaluation, the strength of using CLEval metric is much more apparent. We insert, delete, and replace characters in GT transcriptions to form end-to-end test samples. When using IoU+CRW metric, all H-mean values become 0 since CRW fails to evaluate partially recognized texts. On the other hand, CLEval metric assigns partial scores according to the conditions. In the case of insertion, recall value becomes 1 because predicted transcription contains all GT characters. In the case of deletion, precision value becomes 1 because recognized texts are marked all correct. In the case of replacement, the score is affected by the penalty added to the character that is incorrectly recognized.

Case	Attibutes from CLEval				
	Split	Merge	Miss	Overlap	FP
Original	15	9	0	61	0
Crop 80%	15	9	996	50	0
Crop 60%	12	8	2292	27	0
Crop 40%	9	7	3499	12	96
Split by 2	1014	12	0	60	93
Split by 3	1014	16	0	61	276
Split by 4	1014	19	0	60	581
Overlap 10%	1085	15	0	710	10
Overlap 20%	1093	15	0	1252	2
Overlap 30%	1093	15	0	2020	2

Table 5: Detection attributes from CLEval on ICDAR2013 dataset.

Case	End-to-end Metrics						
	IoU + CRW			CLEval			
	R	P	H	R	P	H	RS
Original	99.6	99.8	99.7	99.7	99.7	99.7	98.9
Insert 1	0.0	0.0	0.0	99.7	84.0	91.2	83.6
Insert 2	0.0	0.0	0.0	99.7	73.1	84.4	73.1
Insert 3	0.0	0.0	0.0	99.7	65.7	79.2	65.6
Delete 1	0.0	0.0	0.0	81.0	99.7	89.4	80.4
Delete 2	0.0	0.0	0.0	63.7	99.6	77.7	63.3
Delete 3	0.0	0.0	0.0	48.0	99.5	64.8	47.8
Replace 1	0.0	0.0	0.0	81.0	81.0	81.0	80.4
Replace 2	0.0	0.0	0.0	64.1	64.2	64.1	63.7
Replace 3	0.0	0.0	0.0	49.9	49.9	49.9	49.7

Table 6: Comparison of end-to-end metrics on toy-example. Some scores are highlighted: Red above 95, Blue below 5.

While performing CLEval end-to-end evaluation, we can also obtain Recognition Score (RS). The RS value is obtained regardless of the detection result by mathematically removing the detection-related terms. A detailed description of RS is provided in the appendix with actual examples.

5. Conclusion

Fair and detailed evaluation of OCR models is needed, and yet, no robust evaluation metric was proposed in the OCR community. The proposed CLEval metric could evaluate text detection, recognition, and end-to-end results. This is done by solving the granularity and correctness issues by performing instance matching and character scoring process. Our metric allows fine assessment, and alleviates qualitative disagreement. We expect researchers and end-users to take advantage of the metric to conduct thorough end-to-end evaluation.

References

[1] Jeonghun Baek, Geewook Kim, Junyeop Lee, Sungrae Park, Dongyoon Han, Sangdoo Yun, Seong Joon Oh, and Hwalsuk Lee. What is wrong with scene text recognition model comparisons? dataset and model analysis. *arXiv preprint arXiv:1904.01906*, 2019. 1

[2] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. Character region awareness for text detection. In *CVPR*, pages 4321–4330. IEEE, 2019. 1

[3] Stefania Calarasanu, Jonathan Fabrizio, and Severine Dubuisson. What is a good evaluation protocol for text localization systems? concerns, arguments, comparisons and solutions. *Image and Vision Computing*, 46:1–17, 2016. 2

[4] Chee Kheng Ch'ng and Chee Seng Chan. Total-text: A comprehensive dataset for scene text detection and recognition. In *ICDAR*, volume 1, pages 935–942. IEEE, 2017. 4

[5] Mark Everingham, S. M. Eslami, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015. 2

[6] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, et al. Icdar 2015 competition on robust reading. In *ICDAR*, pages 1156–1160. IEEE, 2015. 1, 4

[7] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazan Almazan, and Lluis Pere De Las Heras. Icdar 2013 robust reading competition. In *ICDAR*, pages 1484–1493. IEEE, 2013. 4, 7

[8] Chae Young Lee, Youngmin Baek, and Hwalsuk Lee. Tedeval: A fair evaluation metric for scene text detectors. *arXiv preprint arXiv:1907.01227*, 2019. 2, 3

[9] Hong-Seok Lee, Youngmin Yoon, Pil-Hoon Jang, and Chankyu Choi. Popeval: A character-level approach to end-to-end evaluation compatible with word-level benchmark dataset. *arXiv preprint arXiv:1908.11060*, 2019. 2, 5

[10] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. 3

[11] Shangbang Long, Xin He, and Cong Yao. Scene text detection and recognition: The deep learning era. *arXiv preprint arXiv:1811.04256*, 2018. 1

[12] Gonzalo Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001. 3, 5

[13] Diep Thi Ngoc Nguyen and HBLab Jsc. State-of-the-art in action: Unconstrained text detection. 2

[14] Baoguang Shi, Xiang Bai, and Serge Belongie. Detecting oriented text in natural images by linking segments. In *CVPR*, pages 3482–3490. IEEE, 2017. 1

[15] Zhi Tian, Weilin Huang, Tong He, Pan He, and Yu Qiao. Detecting text in natural image with connectionist text proposal network. In *ECCV*, pages 56–72. Springer, 2016. 1

[16] Christian Wolf and Jean-Michel Jolion. Object count/area graphs for the evaluation of object detection and segmentation algorithms. In *ICDAR*, pages 1115–1124. IEEE, 2013. 2

[17] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. East: an efficient and accurate scene text detector. In *CVPR*, pages 2642–2651, 2017. 1