

TedEval: A Fair Evaluation Metric for Scene Text Detectors

Chae Young Lee*, Youngmin Baek*, and Hwalsuk Lee[†]
 Clova AI Research, NAVER Corp.

{cylee.ai, youngmin.baek, hwalsuk.lee}@navercorp.com

Abstract

Despite the recent success of scene text detection methods, common evaluation metrics fail to provide a fair and reliable comparison among detectors. They have obvious drawbacks in reflecting the inherent characteristic of text detection tasks, unable to address issues such as granularity, multiline, and character incompleteness. In this paper, we propose a novel evaluation protocol called TedEval (Text detector Evaluation), which evaluates text detections by an instance-level matching and a character-level scoring. Based on a firm standard rewarding behaviors that result in successful recognition, TedEval can act as a reliable standard for comparing and quantizing the detection quality throughout all difficulty levels. In this regard, we believe that TedEval can play a key role in developing state-of-the-art scene text detectors. The code is publicly available at <https://github.com/clovaai/TedEval>.

1. Introduction

Along with the progress of deep learning, the performance of scene text detectors have remarkably advanced over the past few years [13, 15, 12]. However, providing a fair comparison among such methods is still an ongoing problem, for common metrics fail to reflect the intrinsic nature of text instances [2, 9]. One example is the IoU (Intersection over Union) metric [4]. Adopted from the object detection Pascal VOC, it is designed to detect bounding boxes containing a single object and thus is not suitable to detect text instances consisting of multiple characters. Another approach is DetEval, a metric specifically designed to evaluate text bounding boxes [14]. However, its unreasonably lenient criteria accepts incomplete detections that are prone to fail in the recognition stage. Recently, Liu et al has proposed a new metric named TIoU (Tightness-aware IoU), which adds the tightness penalty to the IoU metric [9]. This approach can be problematic in that it relies on the quality of the Ground Truth (GT), which is often inconsistent.

*Authors contributed equally.

[†]Corresponding author.



(a) Granularity (IoU=0.0) (b) Completeness (DetEval=1.0)

Figure 1: Examples of unfair evaluations. (a) rejected by IoU but should be accepted. (b) accepted by DetEval but should be penalized. Red: GT. Blue: detection.

To solve these issues, a fair evaluation metric for text detectors must account for:

- **Granularity** Annotation tendencies of public datasets vary greatly due to the lack of gold standard on bounding box units. Merging and splitting detection results to match GT should be allowed to a reasonable extent.
- **Completeness** An instance-level match may accept incomplete detections that have missing or overlapping characters. A penalty must be given so that scores reflect whether the detection contains all required characters.

In the proposed metric called *TedEval* (*Text detection Evaluation*), we evaluate text detections via an instance-level matching policy and a character-level scoring policy. Granularity is addressed by non-exclusively gathering all possible matches of one-to-one, one-to-many, and many-to-one. Afterwards, instance-level matches are scored while penalty is given to missing and overlapping characters. To

this end, we use pseudo character centers made from word bounding boxes and their word lengths.

The main contributions of TedEval can be summarized as follows. 1) TedEval is simple, intuitive, and thus easy to use in a wide-range of tasks. 2) TedEval rewards behaviors that are favorable to recognition. 3) TedEval is relatively less affected by the quality of GT annotation. In this regard, we believe that TedEval can play a key role in developing state-of-the-art scene text detectors.

2. Methodology

Our evaluation metric performs the matching and the scoring of detectors through a separate policy. The matching process follows previous metrics, pairing bounding boxes of detections and GTs in an instance-level. On the other hand, the scoring process calculates recall and precision at the character-level without character annotations.

2.1. Matching process

Adopted from DetEval, our evaluation protocol measures area recall and area precision to pair GTs and detections in the manner of not only one-to-one but also one-to-many and many-to-one. To prevent side effects of addressing granularity, TedEval implements three major changes including (1) non-exclusive matching, (2) change in the area recall threshold, and (3) multiline prevention.

The first change directly relates to granularity. Accounting for granularity opens the possibility that a single instance (from either GT or detection) may satisfy multiple matches. DetEval assumes that the first match is the best match, discarding subsequent matches. This causes mismatch among one-to-one and many matches. A simple solution is to neither prioritize nor discard redundant matches. We accept all viable matches, setting M_{ij} of the binary matrix from Table 1 as 1 whenever a match between an instance of GT G_i and detection D_j occurs. When redundant matches occur, M_{ij} is overwritten not accumulated.

Note that the threshold for area recall changed from 0.8

	D_1	...	D_j	Recall
G_1	c_1^1	M_{11}	m_{11}^1	R_1
	c_1^2		m_{11}^2	
	
	$c_1^{l_1}$		$m_{11}^{l_1}$	
G_i
	c_i^1	M_{i1}	m_{i1}^1	R_i
	c_i^2		m_{i1}^2	
	
	$c_i^{l_i}$		$m_{i1}^{l_i}$	
	Precision	P_1	...	P_j

Table 1: Visualization of the matching table with notation.

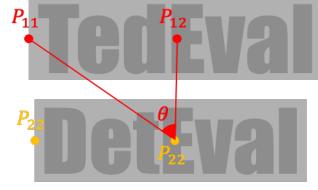


Figure 2: Visualization of multiline computation. The angle θ satisfies Eq. 2 and thus this match is rejected.

to 0.4. This reflects the interaction between the matching process and the scoring process introduced in the next section. Since the latter penalizes the incompleteness of detection to the given GT, namely recall, a lenient threshold in the matching process is fair.

Although our instance matching policy works in a fairly generous way, there is a case when the match itself should be prevented. Multiline is a special case of many match involving multiple lines of texts and must be rejected by identifying it in the matching stage. It involves not only many-to-one match where one detection contains multiple lines but also one-to-many match where multiple detections of different lines are matched to one GT.

As shown in Fig. 2, we use angles to identify multiline. Firstly, define B as a set of bounding boxes in the many side of a many match. Then we compute two pivotal points for all bounding boxes in B by:

$$\begin{aligned} p_{i1} &= \text{mean}(v_{i1}, v_{i4}) \\ p_{i2} &= \text{mean}(v_{i1}, v_{i2}, v_{i3}, v_{i4}) \end{aligned} \quad (1)$$

where v_{ij} means the j th vertex (x, y) of B_i . Note that GT must assume the form of four vertices in a clockwise order starting from the top left point in regard to the orientation of the word.

The angle θ between B_1 and B_2 is then computed by turning from p_{i1} to p_{i2} around p_{i1} . While computing every possible angle among B , we reject the match when any one of it is:

$$|\min(\theta, 180 - \theta)| \geq 45^\circ. \quad (2)$$

The threshold is obtained experimentally.

In addition, instead of selecting pivotal points as points on the edges, p_{i1} is the point on the left edge and p_{i2} is the center point of B_i . This is to make our algorithm robust against the width difference and distance between bounding boxes, which can confuse the magnitude of the angle.

2.2. Scoring process

Based on instance-level matches from Section 2.1, we calculate the recall score of G_i and the precision score of D_j in a character-level. To overcome the lack of character-level annotation in most public datasets, we compute Pseudo Character Centers (PCC) from word-level bounding boxes

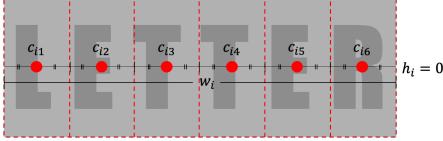


Figure 3: An example of computing PCC of G_i . Red dot: PCC. Red dash: pseudo character box. Grey: G_i .

and their word lengths. As shown in Fig. 3, a set of PCC of G_i is computed by:

$$c_i = \{(x_i + \frac{w_i}{l_i}(k - \frac{1}{2}), y_i + \frac{h_i}{l_i}(k - \frac{1}{2}))\}_{k=1}^{l_i} \quad (3)$$

where x_i and y_i are the x, y coordinates of p_{i1} from Eq. 1, w_i and h_i are Δx and Δy between p_{i1} and the other edge, and l_i is the word length. From the matching table in Table 1, m_{ij} is a binary matrix whose element m_{ij}^k is set to 1 when D_j contains c_i^k .

For recall calculation, we perform a row-wise summation of character matching scores m_i^k :

$$s_i^k = \sum_j m_{ij}^k. \quad (4)$$

Since it is critical that each of the characters is detected only once, the condition of a correct character match is $s_i^k = 1$. Contrastingly, mismatch cases include $s_i^k = 0$ indicating missing characters and $s_i^k > 1$ for overlap characters. The recall R_{G_i} is the number of correct character matches over the text length l_i :

$$R_{G_i} = \frac{|\{s_i^k = 1\}_{k=1}^{l_i}|}{l_i}. \quad (5)$$

On the other hand, the precision P_{D_j} is the number of correct character matches over the sum of text lengths of GTs matched with D_j :

$$P_{D_j} = \frac{\sum_{i \in M_j} \sum_k m_{ij}^k}{\sum_{i \in M_j} l_i} \quad (6)$$

where M_j is $\{x | M_{xj} = 1\}$. Finally, *Recall* and *Precision* can be obtained by

$$\begin{aligned} Recall &= \frac{\sum_{i=1}^{|G|} R_{G_i}}{|G|}, \\ Precision &= \frac{\sum_{j=1}^{|D|} P_{D_j}}{|D|}. \end{aligned} \quad (7)$$

Examples of our scoring process are in the Appendix.

Since scoring occurs column- and row-wise by instance, our scoring policy does not score the same instance multiple times even if it is involved in multiple matches. It can

also differentiate a complete match from a partial match by penalizing missing or overlapping characters. This differs from instance-based scoring, which gives a binary score that does not reflect the completeness of detections.

In addition, TedEval automatically penalizes one-to-many cases, which may abuse the scoring policy by splitting a single word with multiple detections. For example, if a group of detections detect characters of GT once and completely, the precision score of each detection is the number of characters each detected over the length of the GT transcription. Then, the overall precision is 1 over the number of splits. Yet, penalty is not given to many-to-one cases. Examples of scoring many matches are in the Appendix.

3. Experiments

We compared TedEval with DetEval and IoU on two public datasets: ICDAR 2013 Focused Scene Text (IC13) and ICDAR 2015 Incidental Scene Text (IC15). We requested from authors the result file of scene text detectors that frequently appear and are referenced in the literature. Results are shown in Table 2.

Fig. 4 shows the frequency of factors that TedEval tackles as proportions to the number of successful detections. Detectors are sorted right to left from the highest score on TedEval. Granularity counts in average 14% and 14% and completeness counts in average 10% and 16% in IC13 and IC15, respectively. These proportions considerably influence the change of H-mean scores Δ in Table 2 and are the main causes of qualitative discords in previous metrics.

Delving into some of the peaks in Fig. 4, notice that CTPN has the highest granularity frequency in both datasets. As shown in Fig. 5, CTPN has a tendency to detect a single box for an entire line, namely many-to-one. Since such behaviors are not penalized by TedEval, CTPN gains a 6.8 increase in the H-mean score in IC15 from IoU, which does not account for granularity.

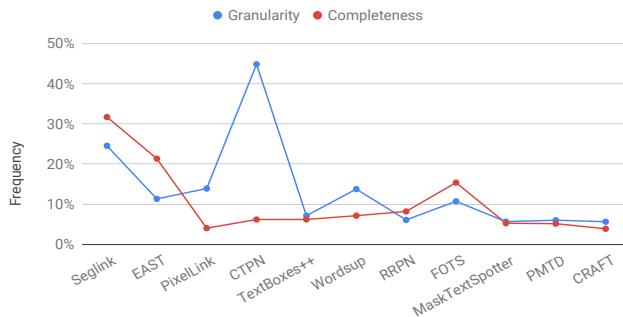
Another peak is from FOTS. As shown in Fig. 5, FOTS often detects a word by splitting it into several parts while causing overlap between such detections. This causes peaks for both granularity and completeness and lowers the recall score. Note that EAST, which proposed the detector architecture of FOTS, shows similar behaviors.

On the contrary, PixelLink and CRAFT have noticeably low completeness counts. They are both segmentation-based detections, which perform well in finding text regions. However, since they connect text regions using link information, they often detect multiline in a single box. In fact, the multiline proportion of PixelLink and CRAFT are one of the highest, 7% and 3%, respectively.

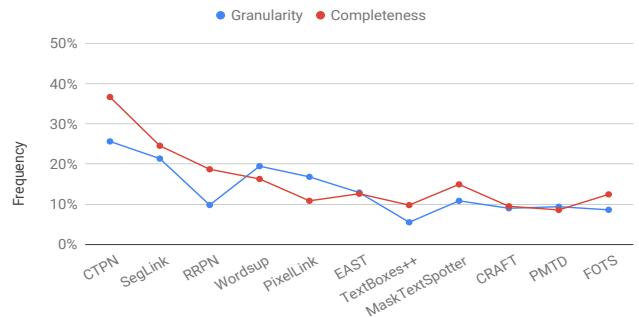
More examples can be seen in the Appendix.

Detector	ICDAR2013						Δ	ICDAR2015						
	DetEval			TedEval				IoU			TedEval			
	R	P	H	R	P	H		R	P	H	R	P	H	
SegLink [12]	60.0	73.9	66.2	65.6	74.9	70.0	3.8	72.9	80.2	76.4	77.1	83.9	80.6	
EAST [15]	70.7	81.6	75.8	77.7	87.1	82.5	6.7	77.2	84.6	80.8	82.5	90.0	86.3	
CTPN [13]	83.0	93.0	87.7	82.1	92.7	87.6	-0.1	51.6	74.2	60.9	85.0	81.1	67.8	
PixelLink [3]	87.5	88.7	88.1	84.0	87.2	86.1	-2.0	83.8	86.7	85.2	85.7	86.1	86.0	
TextBoxes++ [6]	85.6	91.9	88.6	87.4	92.3	90.0	1.4	80.8	89.1	84.8	82.4	90.8	86.5	
WordSup [5]	87.1	92.8	89.9	87.5	92.2	90.2	0.3	77.3	80.5	78.9	83.2	87.1	85.2	
RRPN [11]	87.3	95.2	91.1	89.0	94.2	91.6	0.5	77.1	83.5	80.2	79.5	85.9	82.6	
MaskTextSpotter [10]	88.6	95.0	91.7	90.2	95.4	92.9	1.2	79.5	89.0	84.0	82.5	91.8	86.9	
FOTS [8]	90.4	95.4	92.8	91.5	93.0	92.6	-0.2	87.9	91.9	89.8	89.0	93.4	91.2	
PMTD [7]	92.2	95.1	93.6	94.0	95.2	94.7	1.1	87.4	91.3	89.3	89.2	92.8	91.0	
CRAFT [1]	93.1	97.4	95.2	93.6	96.5	95.1	-0.1	84.3	89.8	86.9	88.5	93.1	90.9	

Table 2: Comparison of evaluation metrics for different detectors. R, P, and H refer to recall, precision, and H-mean. Detectors are sorted from the highest score on DetEval metric. Texts are highlighted in red and blue for rise and fall.



(a) IC13



(b) IC15

Figure 4: Frequency of factors that TedEval tackles counted by predictions. Numbers are represented as proportions to the number of successful detections. Detectors are sorted right to left from the highest score on TedEval.



(a) CTPN ($R : 1.00, P : 1.00$)

(b) FOTS ($R : 0.70, P : 0.65$)

(c) EAST ($R : 0.70, P : 0.45$)

(d) PixelLink ($R : 1.00, P : 1.00$)

Figure 5: Examples of incomplete detections. Numbers in caption indicate recall and precision scores of "SUPERKINGS." Red dot: PCC. Blue: detection.

4. Conclusion

We have proposed a novel evaluation metric for scene text detectors called TedEval, which evaluates text detections by an instance-level matching policy and a character-level scoring policy. It accounts for granularity by adopting DetEval but implements a few changes to prevent subsequent side effects of many matches. A scoring policy uses pseudo character centers to reflect the penalty given to missing and overlapping characters to the final recall and precision score.

Experiments on two public datasets demonstrated that issues TedEval tackles frequently occur in results from state-

of-the-art detectors and that they caused qualitative disagreements in previous metrics. By reflecting such factors, TedEval can provide a fair and reliable evaluation on the state-of-the-art methods in the upper percentile of H-mean scores.

Our future work involves evaluating polygon annotations, where TedEval's logic would be more effective, and making our logic insensitive to the vertex order. This will make TedEval easier to apply in various tasks and lessen the burden of annotators.

Acknowledgements. We would like to thank Yuliang Liu, the author of TIoU, and authors of scene text detectors who kindly provided result files used in our experiments.

References

- [1] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee. Character region awareness for text detection. In *CVPR*, pages 4321–4330. IEEE, 2019. [4](#)
- [2] A. Dangla, E. Puybareau, G. Tochon, and J. Fabrizio. A first step toward a fair comparison of evaluation protocols for text detection algorithms. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 345–350. IEEE, 2018. [1](#)
- [3] D. Deng, H. Liu, X. Li, and D. Cai. Pixellink: Detecting scene text via instance segmentation. In *AAAI*, 2018. [4](#)
- [4] M. Everingham, S. M. Eslami, L. Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015. [1](#)
- [5] H. Hu, C. Zhang, Y. Luo, Y. Wang, J. Han, and E. Ding. Wordsup: Exploiting word annotations for character based text detection. In *ICCV*, 2017. [4](#)
- [6] M. Liao, B. Shi, and X. Bai. Textboxes++: A single-shot oriented scene text detector. *Image Processing*, 27(8):3676–3690, 2018. [4](#)
- [7] J. Liu, X. Liu, J. Sheng, D. Liang, X. Li, and Q. Liu. Pyramid mask text detector. *arXiv preprint arXiv:1903.11800*, 2019. [4](#)
- [8] X. Liu, D. Liang, S. Yan, D. Chen, Y. Qiao, and J. Yan. Fots: Fast oriented text spotting with a unified network. In *CVPR*, pages 5676–5685, 2018. [4](#)
- [9] Y. Liu, L. Jin, Z. Xie, C. Luo, S. Zhang, and L. Xie. Tightness-aware evaluation protocol for scene text detection. In *CVPR*, pages 4321–4330. IEEE, 2019. [1](#)
- [10] P. Lyu, M. Liao, C. Yao, W. Wu, and X. Bai. Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. *arXiv preprint arXiv:1807.02242*, 2018. [4](#)
- [11] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue. Arbitrary-oriented scene text detection via rotation proposals. *IEEE Transactions on Multimedia*, 20(11):3111–3122, 2018. [4](#)
- [12] B. Shi, X. Bai, and S. Belongie. Detecting oriented text in natural images by linking segments. In *CVPR*, pages 3482–3490. IEEE, 2017. [1, 4](#)
- [13] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao. Detecting text in natural image with connectionist text proposal network. In *ECCV*, pages 56–72. Springer, 2016. [1, 4](#)
- [14] C. Wolf and J.-M. Jolion. Object count/area graphs for the evaluation of object detection and segmentation algorithms. In *ICDAR*, pages 1115–1124. IEEE, 2013. [1](#)
- [15] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. East: an efficient and accurate scene text detector. In *CVPR*, pages 2642–2651, 2017. [1, 4](#)

A. Matching matrix

One-to-one						
		Ache		Keta		
		<i>D</i> ₁	<i>D</i> ₂	<i>s</i> _t ^k	<i>Recall</i>	
<i>G</i> ₁	<i>c</i> ₁ ¹	1	1	0	1	4/4
	<i>c</i> ₁ ²		1	0	1	
	<i>c</i> ₁ ³		1	0	1	
	<i>c</i> ₁ ⁴		1	0	1	
<i>G</i> ₂	<i>c</i> ₂ ¹	0	0	1	1	4/4
	<i>c</i> ₂ ²		0	1	1	
	<i>c</i> ₂ ³		0	1	1	
	<i>c</i> ₂ ⁴		0	1	1	
<i>Precision</i>	4/4	4/4				
R	1.0	P	1.0	H	1.0	

Multiline						
		Ache		Keta		
		<i>D</i> ₁	<i>D</i> ₂	<i>s</i> _t ^k	<i>Recall</i>	
<i>G</i> ₁	<i>c</i> ₁ ¹	0	0	0	0	0/4
	<i>c</i> ₁ ²		0	0	0	
	<i>c</i> ₁ ³		0	0	0	
	<i>c</i> ₁ ⁴		0	0	0	
<i>G</i> ₂	<i>c</i> ₂ ¹	0	0	0	0	0/4
	<i>c</i> ₂ ²		0	0	0	
	<i>c</i> ₂ ³		0	0	0	
	<i>c</i> ₂ ⁴		0	0	0	
<i>Precision</i>	0/8					
R	0.0	P	0.0	H	0.0	

One-to-many						
		Ache		Keta		
		<i>D</i> ₁	<i>D</i> ₂	<i>s</i> _t ^k	<i>Recall</i>	
<i>G</i> ₁	<i>c</i> ₁ ¹	1	1	0	1	8/8
	<i>c</i> ₁ ²		1	0	1	
	<i>c</i> ₁ ³		1	0	1	
	<i>c</i> ₁ ⁴		1	0	1	
	<i>c</i> ₁ ⁵		0	1	1	
	<i>c</i> ₁ ⁶		0	1	1	
	<i>c</i> ₁ ⁷		0	1	1	
	<i>c</i> ₁ ⁸		0	1	1	
	<i>Precision</i>		4/8	4/8		
R	1.0	P	0.5	H	0.66	

Many-to-one						
		Ache		Keta		
		<i>D</i> ₁	<i>D</i> ₂	<i>s</i> _t ^k	<i>Recall</i>	
<i>G</i> ₁	<i>c</i> ₁ ¹	1	1	1	1	4/4
	<i>c</i> ₁ ²		1	1	1	
	<i>c</i> ₁ ³		1	1	1	
	<i>c</i> ₁ ⁴		1	1	1	
<i>G</i> ₂	<i>c</i> ₂ ¹	1	1	1	1	4/4
	<i>c</i> ₂ ²		1	1	1	
	<i>c</i> ₂ ³		1	1	1	
	<i>c</i> ₂ ⁴		1	1	1	
<i>Precision</i>	8/8					
R	1.0	P	1.0	H	1.0	

Overlap characters						
		Ache		Keta		
		<i>D</i> ₁	<i>D</i> ₂	<i>s</i> _t ^k	<i>Recall</i>	
<i>G</i> ₁	<i>c</i> ₁ ¹	1	1	0	1	6/8
	<i>c</i> ₁ ²		1	0	1	
	<i>c</i> ₁ ³		1	0	1	
	<i>c</i> ₁ ⁴		1	1	2	
	<i>c</i> ₁ ⁵		1	1	2	
	<i>c</i> ₁ ⁶		0	1	1	
	<i>c</i> ₁ ⁷		0	1	1	
	<i>c</i> ₁ ⁸		0	1	1	
<i>Precision</i>	6/8	6/8				
R	0.75	P	0.75	H	0.75	

Missing characters						
		Ache		Keta		
		<i>D</i> ₁	<i>D</i> ₂	<i>s</i> _t ^k	<i>Recall</i>	
<i>G</i> ₁	<i>c</i> ₁ ¹	1	0	0	0	4/8
	<i>c</i> ₁ ²		0	0	0	
	<i>c</i> ₁ ³		1	1	1	
	<i>c</i> ₁ ⁴		1	1	1	
	<i>c</i> ₁ ⁵		1	1	1	
	<i>c</i> ₁ ⁶		1	1	1	
	<i>c</i> ₁ ⁷		0	0	0	
	<i>c</i> ₁ ⁸		0	0	0	
<i>Precision</i>	4/8					
R	0.5	P	0.5	H	0.5	

Figure 6: Examples of scoring in various cases.

B. Detection results



(a) PixelLink ($R : 1.00, P : 0.80$) (b) WordSup ($R : 1.00, P : 1.00$) (c) TB++ ($R : 1.00, P : 0.86$) (d) PMTD ($R : 1.00, P : 1.00$)

Figure 7: Granularity



(a) CTPN ($R : 0.75, P : 1.00$) (b) PixelLink ($R : 0.90, P : 0.90$) (c) WordSup ($R : 0.25, P : 0.50$) (d) MaskTS ($R : 1.00, P : 1.00$)

Figure 8: Completeness



(a) CTPN ($R : 0.83, P : 1.00$) (b) PixelLink ($R : 0.83, P : 0.83$) (c) MaskTS ($R : 1.00, P : 1.00$) (d) CRAFT ($R : 0.83, P : 1.00$)

Figure 9: Multiline



(a) TB++ ($R : 0.97, P : 0.97$) (b) WordSup ($R : 0.80, P : 1.00$) (c) FOTS ($R : 0.92, P : 0.83$) (d) CRAFT ($R : 1.00, P : 1.00$)

Figure 10: Text-in-text