AE TextSpotter: Learning Visual and Linguistic Representation for Ambiguous Text Spotting

Wenhai Wang[1], Xuebo Liu[2], Xiaozhong Ji[1], Enze Xie[3], Ding Liang[2]
ZhiBo Yang[4], Tong Lu[1,✉], Chunhua Shen[5], and Ping Luo[3]

[1]National Key Lab for Novel Software Technology, Nanjing University
[2]SenseTime Research   [3]The University of Hong Kong
[4]Alibaba-Group   [5]The University of Adelaide
`wangwenhai362@smail.nju.edu.cn`, {`liuxuebo, liangding`}`@sensetime.com`,
`shawn_ji@163.com`, `xieenze@hku.hk`, `zhibo.yzb@alibaba-inc.com`,
`lutong@nju.edu.cn`, `chunhua.shen@adelaide.edu.au`, `pluo@cs.hku.hk`

Abstract. Scene text spotting aims to detect and recognize the entire word or sentence with multiple characters in natural images. It is still challenging because ambiguity often occurs when the spacing between characters is large or the characters are evenly spread in multiple rows and columns, making many visually plausible groupings of the characters (*e.g.* "BERLIN" is incorrectly detected as "BERL" and "IN" in Fig. 1(c)). Unlike previous works that merely employed visual features for text detection, this work proposes a novel text spotter, named Ambiguity Eliminating Text Spotter (AE TextSpotter), which learns both visual and linguistic features to significantly reduce ambiguity in text detection. The proposed AE TextSpotter has three important benefits. 1) The linguistic representation is learned together with the visual representation in a framework. To our knowledge, it is the first time to improve text detection by using a language model. 2) A carefully designed language module is utilized to reduce the detection confidence of incorrect text lines, making them easily pruned in the detection stage. 3) Extensive experiments show that AE TextSpotter outperforms other state-of-the-art methods by a large margin. For example, we carefully select a set of extremely ambiguous samples from the IC19-ReCTS dataset, where our approach surpasses other methods by more than 4%.

Keywords: Text Spotting, Text Detection, Text Recognition, Text Detection Ambiguity

1 Introduction

Text analysis in unconstrained scene images like text detection and text recognition is important in many applications, such as document recognition, license plate recognition, and visual question answering based on texts. Although previous works [27, 33, 19, 20, 12, 13] have acquired great success, there are still many challenges to be solved.
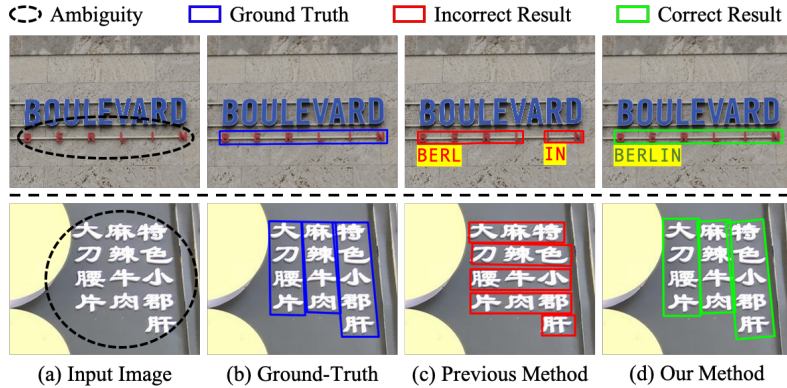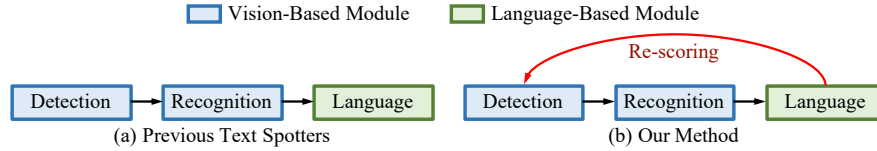
Fig. 1. The detection results of different methods in cases of ambiguity. Top row: an example of large character spacing. Bottom row: an example of juxtaposed text lines. (a) are original images. (b) are ground-truths. (c) are detection results of Mask TextSpotter [13]. (d) are our detection results.

This work addresses one of the important challenges, which is reducing the ambiguous bounding box proposals in scene text detection. These ambiguous proposals widely occur when the spacing of the characters of a word is large or multiple text lines are juxtaposed in different rows or columns in an image. For example, as shown in Fig. 1(c)(d), the characters in an image can be grouped into multiple visually plausible words, making the detection results ambiguous and significantly hindering the accuracy of text detectors and text spotters.

Existing text spotters [19, 13, 22, 7] typically follow a pipeline as shown in Fig. 2(a), where a text recognition module is stacked after a text detection module, and linguistic knowledge (*e.g.* lexicon) is employed for text recognition. However, their language-based modules are isolated from the detection modules, which only detect text lines by learning visual features, leading to ambiguous proposals and hindering the accuracy of the entire systems. As shown in Fig. 1(c), these vision-based text detectors are insufficient to detect text lines correctly in ambiguous samples (*i.e.* horizontal bounding box versus vertical bounding box in this example).

In contrast, unlike existing text detectors that only utilize vision knowledge, this work incorporates linguistic knowledge into text detection by learning linguistic representation to reduce ambiguous proposals. As shown in Fig. 2(b), the re-scoring step is not only the main difference between our approach and the previous methods, but also the key step to remove ambiguity in text detection. Intuitively, without a natural language model, it is difficult to identify the correct text line in ambiguous scenarios. For example, prior art [13] detected incorrect text lines as shown in Fig.1(c), neglecting the semantic meaning of text lines. In our approach, the knowledge of natural language can be used to reduce errors as shown in Fig. 1(d), because the text content may follow the distribution of natu-

Vision-Based Module   Language-Based Module

Re-scoring

Detection → Recognition → Language   Detection → Recognition → Language

(a) Previous Text Spotters   (b) Our Method

Fig. 2. Comparisons between (a) previous text spotters and (b) our AE TextSpotter. Different from previous methods [19, 13, 22, 7] where the modules of text detection and linguistic knowledge are isolated, our method lifts the language module ahead in text detection to re-score the detection results by learning linguistic representation.

ral language if the detected text line is correct, otherwise the text content is out of the distribution of natural language. Therefore, the linguistic representation can help refine the detection results.

This paper proposes a novel framework called Ambiguity Eliminating Text Spotter (AE TextSpotter) as shown in Fig. 2(b), to solve the ambiguity problem in text detection. The rationale of our method is to re-score the detection results by using linguistic representation, and eliminate ambiguity by filtering out low-scored text line proposals. Specifically, AE TextSpotter consists of three important components, including detection, recognition and rescoring.

In the detection step, we build a text detection module (TDM) to provide candidates of text lines with high recall, in order to avoid missed detection. In this step, the number of candidates is very large because of the high-recall rate. In the step of recognition, we carefully design a character-based recognition module (CRM), to efficiently recognize the candidate text lines. In the re-scoring step, we propose a language module (LM) that can learn linguistic representation to re-score the candidate text lines and to eliminate ambiguity, making the text lines that correspond to natural language have higher scores than those not. The three steps above complement each other, enabling the AE TextSpotter to successfully detect and recognize text lines even in extremely challenging scenarios. As shown in Fig. 1(d), AE TextSpotter can correctly detect text lines when multiple vertical and horizontal text lines are presented.

Our main **contributions** are summarized as follows. **1)** The AE TextSpotter identifies the importance of linguistic representation for text detection, in order to solve the text detection ambiguity problem. To our knowledge, this is the first time to learn both visual features and linguistic features for text detection in a framework. **2)** We design a novel text recognition module, called the character-based recognition module (CRM), to achieve fast recognition of numerous candidate text lines. **3)** Extensive experiments on several benchmarks demonstrate the effectiveness of AE TextSpotter. For example, AE TextSpotter achieves state-of-the-art performance on both text detection and recognition on IC19-ReCTS [32]. Moreover, we construct a hard validation set for benchmarking **T**ext **D**etection **A**mbiguity, termed TDA-ReCTS, where AE TextSpotter achieves the F-measure of 81.39% and the 1-NED of 51.32%, outperforming its counterparts by 4.01% and 4.65%.

2   Related Work

Scene text detection has been a research hotspot in computer vision for a long period. Methods based on deep learning have become the mainstream of scene text detection. Tian *et al.* [27] and Liao *et al.* [15] successfully adopted the framework of object detection into text detection and achieved good performance on horizontal text detection. After that, many works [33, 24, 4, 14, 17] took the orientation of text lines into consideration and make it possible to detect arbitrary-oriented text lines. Recently, curved text detection attracted increasing attention, and segmentation-based methods [20, 12, 30, 31] achieved excellent performances over the curved text benchmarks. These methods improve the performance of text detection to a high level, but none of them can deal with the ambiguity problem in text detection. In this work, we introduce linguistic features in the text detection module to solve the text detection ambiguity problem.

Scene text recognition targets to decode a character sequence from variable-length text images. CRNN [25] introduced CTC [8] in a text recognition model, and made the model trained end-to-end. Following this pipeline, other CTC-based methods [18, 28] were proposed and achieved significant improvement in text recognition. Another direction of text recognition is attention-based encoder and decoder framework [26, 2, 1], which focuses on one position and predict the corresponding character at each time step. These attention-based methods have achieved high accuracy on various benchmarks. However, most existing methods work slowly when recognizing a large number of candidate text lines in an image. In this paper, we design a character-based recognition module to solve this problem.

Scene text spotting aims to detect and recognize text lines simultaneously. Li *et al.* [11] proposed a framework for horizontal text spotting, which contains a text proposal network for text detection and an attention-based method for text recognition. FOTS [19] presented a method for oriented text spotting by new RoIRotate operation, and significantly improved the performance by jointly training detection and recognition tasks. Mask TextSpotter [13] added character-level supervision to Mask R-CNN [9] to simultaneously identity text lines and characters, which is suitable for curved text spotting. Qin *et al.* [22] also developed a curved text spotter based on Mask R-CNN, and used a RoI masking operation to extract the feature for curved text recognition. However, in these methods, linguistic knowledge (*e.g.* lexicon) is only used in post-processing, so it is isolated from network training. Different from them, the linguistic representation in our method is learned together with the visual representation in a framework, making it possible to eliminate ambiguity in text detection.

3   Proposed Method

3.1   Overall Architecture

Fig. 3 shows the overall architecture of AE TextSpotter, which consists of two vision-based modules and one language-based module, namely, the text detec-
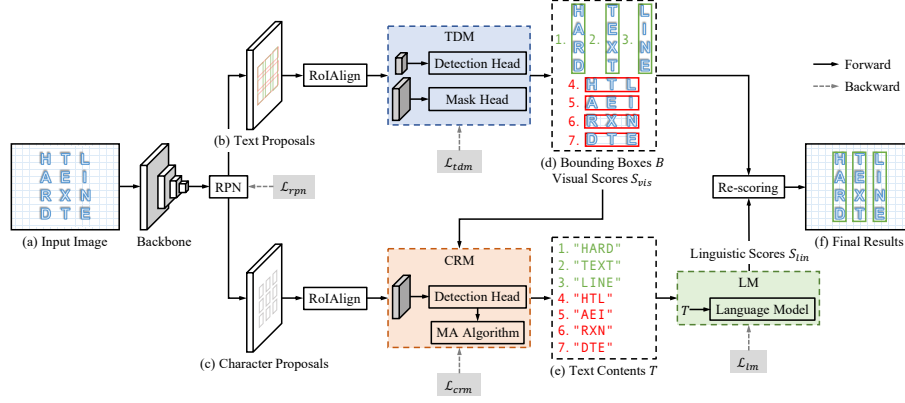
Fig. 3. The overall architecture of AE TextSpotter, which has three key modules: 1) Text Detection Module (TDM); 2) Character-based Recognition Module (CRM); 3) Language Module (LM).

tion module (TDM), the character-based recognition module (CRM), and the language module (LM). Among these modules, TDM and CRM aim to detect the bounding boxes and recognize the content of candidate text lines; and LM is applied to lower the scores of incorrect text lines by utilizing linguistic features, which is the key module to remove ambiguous samples.

In the forward phase, we firstly feed the input image to the backbone network (*i.e.*, ResNet-50 [10] + FPN [16]) and Region Proposal Network (RPN) [23], to produce text line proposals and character proposals. Secondly, based on text line proposals, TDM predicts tight bounding boxes B (*i.e.*, rotated rectangles or polygons) and visual scores S_{vis} for candidate text lines (see Fig. 3(d)). Thirdly, using character proposals and bounding boxes B as input, CRM recognizes the text contents T of candidate text lines (see Fig. 3(e)). After that, we feed the text contents T to LM, to predict the linguistic scores S_{lin} for candidate text lines. Then we combine the visual scores S_{vis} and linguistic scores S_{lin} into the final scores S as Eqn 1.

$$S = \lambda S_{vis} + (1 - \lambda)S_{tex}, \tag{1}$$

where λ is used to balance S_{vis} and S_{tex}. Finally, on the basis of the final scores S, we select the final bounding boxes and text contents of text lines (see Fig. 3(f)), by using NMS and removing low-score text lines.

The training process of AE TextSpotter is divided into two stages: 1) training the backbone network, RPN, TDM, and CRM end-to-end; 2) training LM with the predicted text lines and text contents. In the first stage, we fix the weights of LM, and optimize the backbone network, RPN, TDM, and CRM by a joint loss of $\mathcal{L}_{rpn}$, $\mathcal{L}_{tdm}$, and $\mathcal{L}_{crm}$. In the second stage, we fix the weights of the backbone network, RPN, TDM, and CRM, and optimize LM by loss function $\mathcal{L}_{lm}$.
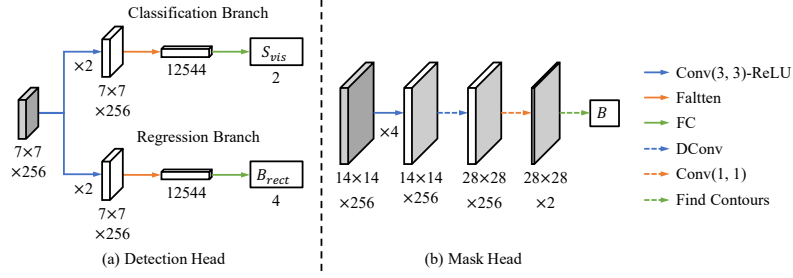
Fig. 4. The details of the detection head and the mask head in the text detection module (TDM). "Conv(3, 3)", "FC" and "DConv" represent the convolution layer with the kernel size of 3×3, fully connected layer, and deconvolution layer, respectively. "×2" denotes a stack of two consecutive convolution layers.

3.2 Text Detection Module

The text detection module (TDM) aims to detect the candidate text lines with extremely high recall. Due to the requirement of high recall, TDM needs to be implemented by a regression-based method, whose recall can be controlled by the classification score threshold and the NMS threshold. In this work, following [31, 13, 22], TDM is modified based on Mask R-CNN [9]. For each text line, it predicts an axis-aligned rectangular bounding box and the corresponding segmentation mask.

As shown in Fig. 4, there are a detection head and a mask head in TDM. Firstly, we use RoIAlign [9] to extract feature patches (7×7×256) for text line proposals predicted by RPN. Secondly, we feed feature patches to the classification and regression branches in the detection head, to predict visual scores S_{vis} and rectangular bounding boxes B_{rect} of text lines. Then, the new feature patches (14×14×256) are extracted for rectangular bounding boxes B_{rect}. After that, using the new feature patches as input, the mask head predicts masks of text lines, and the mask contours are tight bounding boxes B. Finally, we use a loose score threshold Thr_{score} and a loose NMS threshold Thr_{nms} to filter out redundant boxes in tight bounding boxes B, and the rest are candidate text lines.

3.3 Character-based Recognition Module

The character-based recognition module (CRM) is applied to efficiently recognize a large number of candidate text lines produced by TDM. CRM includes a detection head and a match-assemble (MA) algorithm. The detection head runs only once for each input image to detect and classify characters; and the proposed MA algorithm is used to group characters into text line transcriptions, which is rule-based and very fast. Therefore the CRM is very suitable for recognizing numerous candidate text lines predicted by TDM.
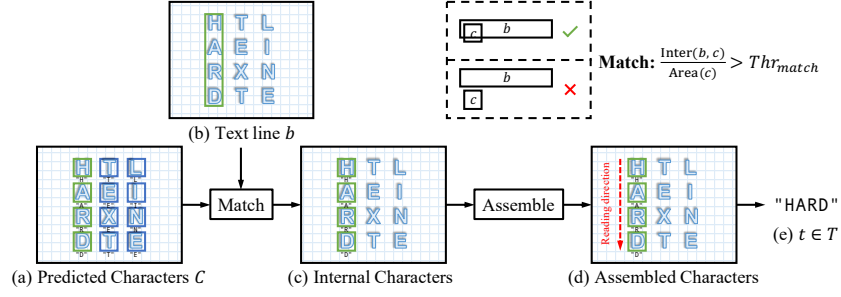
Fig. 5. The details of match-assemble (MA) algorithm. "Match" is the operation of matching character boxes with a text line boxes. "Assemble" is the operation of assembling the internal characters to a text line transcription.

The detection head of CRM follows a pipeline similar to the detection head of TDM (see Fig. 4 (a)). Due to the large number of character categories (about 3,600 in IC19-ReCTS), the detection head of CRM needs a better feature patch as input and a deeper classification branch. Therefore, in the detection head of CRM, the size of feature patches extracted by RoIAlign is $14\times14\times256$, and we change the number of convolutional layers in the classification branch to 4. The output of the detection head is rectangular bounding boxes and contents of characters.

After character detection and classification, we propose a match-assemble (MA) algorithm to obtain text contents. The procedure of the MA algorithm is presented in Fig. 5. For each bounding box $b \in B$ of candidate text line, we firstly **match** the predicted characters (see Fig. 5(a)) with the candidate text line b by using Eqn. 2, and get the internal characters C_{in} (see Fig. 5 (c)) for the candidate text line b.

$$C_{in} = \{c \mid \frac{\text{Inter}(c, b)}{\text{Area}(c)} > Thr_{match}, c \in C\} \tag{2}$$

Here, $\text{Inter}(\cdot)$, $\text{Area}(\cdot)$ and C are the intersection area of two boxes, the area of a box, and bounding boxes of detected characters, respectively. Secondly, we **assemble** the internal characters C_{in} according to their center positions. Specifically, we arrange the characters from left to right, when the width of the external horizontal rectangle of the candidate text line is larger than its height, or else we arrange the characters from top to down. After the arrangement, we concatenate character contents into the text content T of the candidate text line.

3.4   Language Module

The language module (LM) is a language-based module that can re-score candidate text lines according to their contents. Specifically, LM predicts a linguistic
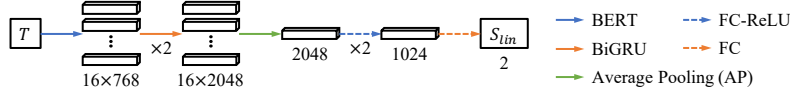
Fig. 6. The details of Language Module (LM). "BiGRU", and "FC" represent the binary input gated recurrent unit [3], and fully connected layer, respectively. "×2" denotes a stack of two consecutive layers.

score to determine whether the text content follows the distribution of natural language.

The pipeline of LM is presented in Fig. 6. At first, we use BERT [6] , a widely-used pre-trained model in NLP, to extract sentence vectors (16×768) for the input text content T. Here, 16 is the maximum length of text content, and we fill 0 for text lines whose length is less than 16. Then we feed sentence vectors to a classification network composed of BiGRU, AP, and FC , to predict the linguistic scores S_{lin} of the input text contents. Here, BiGRU, AP, and FC represent binary input gated recurrent unit [3], average pooling layer, and fully connected layer, respectively.

The label generation for LM is based on Intersection over Union (IoU) metric. Concretely, for every candidate text line, we consider the label of its text content to be positive if and only if there is a matched ground-truth whose IoU with the candidate text line is greater than a threshold IoU_{pos}.

3.5   Loss Function

As mentioned in Sec. 3.1, the training of AE TextSpotter has two stages. Correspondingly, the loss function has two parts: 1) the visual loss function $\mathcal{L}_{vis}$; 2) the linguistic loss function $\mathcal{L}_{lm}$.

The visual loss function $\mathcal{L}_{vis}$ is used to optimize the backbone network, RPN, TDM, and CRM. It is a multi-task loss function that can be defined as Eqn. 3.

$$\mathcal{L}_{vis} = \mathcal{L}_{rpn} + \mathcal{L}_{tdm} + \mathcal{L}_{crm}, \tag{3}$$

where $\mathcal{L}_{rpn}$ is loss function of RPN as defined in [23]. $\mathcal{L}_{tdm}$ and $\mathcal{L}_{crm}$ are loss functions of TDM and CRM respectively, which can be formulated as Eqn. 4 and Eqn. 5.

$$\mathcal{L}_{tdm} = \mathcal{L}_{tdm}^{cls} + \mathcal{L}_{tdm}^{box} + \mathcal{L}_{tdm}^{mask}. \tag{4}$$

$$\mathcal{L}_{crm} = \mathcal{L}_{crm}^{cls} + \mathcal{L}_{crm}^{box}. \tag{5}$$

Here, $\mathcal{L}_{tdm}^{cls}$ and $\mathcal{L}_{crm}^{cls}$ are loss functions of classification branches in TDM and CRM respectively, which are implemented by cross-entropy losses. $\mathcal{L}_{tdm}^{box}$ and $\mathcal{L}_{crm}^{box}$ are loss functions of regression branches in TDM and CRM respectively, which are implemented by smooth L_1 [3]. $\mathcal{L}_{tdm}^{mask}$ is the loss function of mask head in TDM, which is the same as the mask loss in [9].

The linguistic loss function $\mathcal{L}_{lm}$ is used to optimize LM. Actually, LM only involves a task of classifying the correct and incorrect text lines in candidates, so $\mathcal{L}_{lm}$ is implemented by a cross-entropy loss.

4   Experiment

4.1   Datasets

AE TextSpotter needs character-level annotations in the training phase. In addition text detection ambiguity is relatively rare in English due to its writing habit. Therefore, we evaluate our method on a multi-language dataset with character-level annotations.

ICDAR 2019 ReCTS (IC19-ReCTS) [32] is a newly-released large-scale dataset that includes 20,000 training images and 5,000 testing images, covering multiple languages, such as Chinese, English and Arabic numerals. The images in this dataset are mainly collected in the scenes with signboards, All text lines and characters in this dataset are annotated with bounding boxes and transcripts.

TDA-ReCTS is a validation set for benchmarking text detection ambiguity, which contains 1,000 ambiguous images selected from the training set of IC19-ReCTS. To minimize the impact of subjective factors, we designed rules to pick out images with the case of large character spacing or juxtaposed text lines, and then randomly sample 1,000 images among them as the validation set.

When selecting ambiguous images, an image is regarded as a sample with large character spacing, if at least one text line in the image has large character spacing. In addition, an image is treated as a sample with juxtaposed text lines, if there is a pair of text lines aligned with the top, bottom, left or right direction, and their characters have similar scales. More details about the both rules can be found in Appendix A.1.

4.2   Implementation Details

In the backbone network, the ResNet50 [10] is initialized by the weights pre-trained on ImageNet [5]. In RPN, the aspect ratios of text line anchors are set to $\{1/8, 1/4, 1/2, 1, 2, 4, 8\}$ to match the extreme aspect ratios of text lines, and the aspect ratios of character anchors are set to $\{1/2, 1, 2\}$. All other settings in RPN are the same as in Mask R-CNN. In TDM, the classification branch makes a binary classification of text and non-text, and the two post-processing thresholds (*i.e.*, Thr_{score} and Thr_{nms}) in the regression branch are set to 0.01 and 0.9, respectively. In CRM, the match threshold Thr_{match} is set to 0.3. In LM, candidate text lines are classified into correct and incorrect ones., and the training label threshold IoU_{pos} is set to 0.8. When combining the visual and linguistic scores, the balance weight λ in Eqn. 1 is set to 0.7. In the post-processing, the NMS threshold is set to 0.1 and the classification score threshold is set to 0.6.

As mentioned in Sec. 3.1, the training of AE TextSpotter is divided into two phases. In the first phase, we train backbone network, RPN, TDM, and CRM with batch size 16 on 8 GPUs for 12 epochs, and the initial learning rate is set to 0.02 which is divided by 10 at 8 epoch and 11 epoch. In the second phase, we fix the weights of the backbone network, RPN, TDM, and CRM, and train LM with batch size 16 on 8 GPUs for 12 epochs, and the initial learning rate is set

Table 1. The proportion of text lines with the problem of text detection ambiguity.

Type	IC19-ReCTS Training Set		TC-ReCTS	
	Number	Proportion	Number	Proportion
Large Character Spacing	1,241	1.14%	589	7.07%
Juxtaposed Text Lines	5,379	4.94%	1,615	19.39%
Union of two Categories	6,509	5.97%	2,148	25.80%
All	108,963	-	8,327	-

to 0.2 which is divided by 10 at 8 epoch and 11 epoch. We optimize all models using stochastic gradient descent (SGD) with a weight decay of 1×10^{-4} and a momentum of 0.9.

In the training phase, we ignore the blurred text regions labeled as "DO NOT CARE" in all datasets; and apply random scale, random horizontal flip, and random crop on training images.

4.3   Ablation Study

The proportion of ambiguous text lines.   Text detection ambiguity is caused by large character spacing and juxtaposed text lines. To count the proportion of the two problematic cases, we use the rules mentioned in Sec. 4.1 to count the ambiguous text lines in the IC19-ReCTS training set and TDA-ReCTS. As shown in Table 1, ambiguous text lines occupy 5.97% of total text lines in the IC19-ReCTS training set, which reveals that text detection ambiguity is not rare in the natural scene. Moreover, the proportion of ambiguous text lines in TDA-ReCTS reaches 25.80%. Although the proportion in TDA-ReCTS is much larger than in the IC19-ReCTS training set, the normal text lines are still in the majority.

The high recall of the text detection module. To verify the high recall of the text detection module (TDM), we compare the recall and the number of candidate text lines per image on TDA-ReCTS, under different visual score thresholds Thr_{score} and the NMS thresholds Thr_{nms}. As shown in Table. 2, when $Thr_{score} = 0.0$ and $Thr_{nms} = 1.0$, the recall of TDM reaches the upper bound of 0.971. However, in this case, the number of candidate text lines per image is 971.5, which is too large to be recognized. When $Thr_{score} = 0.01$ and $Thr_{nms} = 0.9$, the recall of TDM is 0.969, slightly lower than the upper bound, but the number of candidate text lines per image is reduced to 171.7. If we use common post-processing thresholds (*i.e.*, $Thr_{score} = 0.5$ and $Thr_{nms} = 0.5$), the recall of TDM is only 0.852, which will result in many missed detection of text lines. Therefore, in experiments, the values of Thr_{score} and Thr_{nms} are set to 0.01 and 0.9 by default, which can achieve a high recall without too many candidate text lines.

The effectiveness of the character-based recognition module. The character-based recognition module (CRM) aims to fast recognize numerous candidate text lines predicted by TDM. To test the speed of CRM, we compare the time cost per text line of CRM and mainstream recognizers (*e.g.* CRNN [25] and

Table 2. The recall of TDM and the number of candidate text lines per image under different post-processing thresholds.

Thr_{score}	Thr_{nms}	Recall	Number of Candidates
0	1	0.971	971.5
0.01	0.9	0.969	171.7
0.5	0.5	0.852	11.1

Table 3. The time cost per image and 1-NED of different recognizers on the IC19-ReCTS test set.

Method	Time Cost	1-NED
CRNN [25]	966.7 ms	68.32
ASTER [26]	4828.2 ms	58.49
CRM (ours)	122.3 ms	71.81

(a) w/o LM

(b) w/ LM
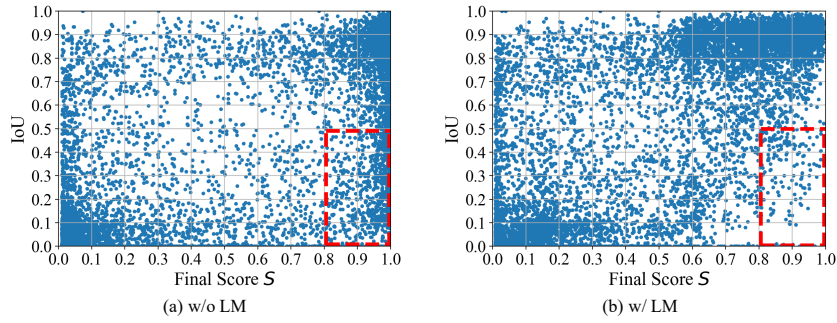
Fig. 7. The distributions of detection results predicted by models with and without LM. The points in **red dotted boxes** are false positives with high scores.

ASTER [26]). For a fair comparison, we run these methods on 1080Ti GPU and report their time cost per image on the IC19-ReCTS test set. Note that, the input text images of CRNN and ASTER are scaled to a height of 32 when testing. In AE TextSpotter, the recognition module is used to recognize the candidate text lines, whose number reaches **171.7 per image**. Table. 3 shows the speed of different methods. CRNN and ASTER cost 966.7 ms and 4828.2 ms respectively to recognize **all candidate text lines** in each image. The time cost of CRM on each image is only 122.3 ms, so CRM runs much faster than CRNN and ASTER. In addition, using the same detection result predicted by AE TextSpotter, our CRM can achieve the 1-NED of 71.81 on the test set of IC19-ReCTS, which is better than CRNN and ASTER.

The effectiveness of the language module. To investigate the effectiveness of the language module (LM), we compare the performance of the models with and without LM on TDA-ReCTS. For a fair comparison, we build a model without LM by removing LM from AE TextSpotter in the training phase and setting λ in Eqn. 1 to 1.0 in the testing phase. As shown in Table 4, on TDA-ReCTS, the model with LM obtains the F-measure of 81.39 and the 1-NED of 51.32%, significantly surpassing the model without LM by 3.46% and 3.57%.

In addition, we analyze the quality of the detection results predicted by the models with and without LM on TDA-ReCTS. Generally, the quality of a bounding box can be measured by its IoU with the matching ground truth, and a bounding box is considered to be correct if its IoU exceeds 0.5. As shown in

Fig. 7 (a), the detection results of the model without LM have many incorrect results (IoU < 0.5), even though their final scores are greater than 0.8. However, this problem is alleviated in the model with LM (see Fig. 7 (b)), where there are few detection results with high scores but low IoU. These experiment results demonstrate the power of LM.

4.4   Comparisons with State-of-the-Art Methods

To show the effectiveness of the proposed AE TextSpotter, we compare it with state-of-the-art methods on TDA-ReCTS and IC19-ReCTS test set.

On TDA-ReCTS, we compare the proposed AE TextSpotter with some representative methods. All results are obtained by using authors provided or officially released codebases, and these methods are trained on the 19,000 training images in IC19-ReCTS (except 1,000 images in TDA-ReCTS). In the testing phase, we scale the short side of images to 800, and evaluate the performance of these methods on TDA-ReCTS by the evaluation metric defined in [32]. Table 4 shows the performance of these methods. The proposed AE TextSpotter achieves the F-measure of 81.39% and the 1-NED of 51.32%, outperforming the counterparts by at least 4.01% and 4.65% respectively. Here, 1-NED denotes normalized edit distance, a recognition metric defined in [32]. Note that, the proportion of ambiguous text lines in TDA-ReCTS is only 25.80%, and we believe the improvement will be more significant if the proportion is larger.

On the IC19-ReCTS test set, we train models on the complete training set of IC19-ReCTS. As shown in Table 5, our AE TextSpotter achieves the F-measure of 91.80% and the 1-NED of 71.81%, surpassing other methods. Moreover, when we enlarge the scale of input images (short size 1000), the F-measure and 1-NED of our method can be boosted to 91.86% and 73.12%. Without extra datasets, the single-scale detection performance of our method is comparable to the methods that use extra datasets, multi-scale testing, and model ensemble (91.86% *v.s.* 92.99%). Unlike the detection task, the multi-language recognition task relies more heavily on extra datasets. Therefore, our recognition performance is lower than the methods using extra datasets, multi-scale testing, and ensemble.

These results demonstrate that the proposed AE TextSpotter can correctly detect and recognize multi-language text lines, even in ambiguous scenarios. Some qualitative results on TDA-ReCTS are shown in Fig. 8.

4.5   Time Cost Analysis of AE TextSpotter

We analyze the time cost of all modules in AE TextSpotter. Specifically, we evaluate our methods on images in TDA-ReCTS and calculate the average time cost per image. All results are tested by PyTorch [21] with the batch size of 1 on one 1080Ti GPU in a single thread. Table 6 shows the time cost of backbone, TDM, CRM, LM, and post-processing respectively. Among them, the detection module takes up most of the execution time, because it needs to predict the mask for every candidate text line. The entire framework can run 1.3 FPS when the short size of input images are 800.

Table 4. The single-scale results on TDA-ReCTS. "P", "R", "F" and "1-NED" mean the precision, recall, F-measure, and normalized edit distance [32], respectively.

Method	Venue	Backbone	TDA-ReCTS			
			P	R	F	1-NED
EAST [33]	CVPR'17	VGG16	70.58	61.59	65.78	-
PSENet [29]	CVPR'19	ResNet50	72.14	65.53	68.68	-
FOTS [19]	CVPR'18	ResNet50	68.20	70.00	69.08	34.87
Mask TextSpotter [13]	TPAMI	ResNet50	80.07	74.86	77.38	46.67
Detection Only	-	ResNet50	79.58	74.93	77.19	-
AE TextSpotter w/o LM	-	ResNet50	79.18	76.72	77.93	47.74
AE TextSpotter	-	ResNet50	**84.78**	**78.28**	**81.39**	**51.32**

Table 5. The single-scale results on the IC19-ReCTS test set. "P", "R", "F" and "1-NED" represent the precision, recall, F-measure, and normalized edit distance, respectively. "*" denotes the methods in competition [32], which use extra datasets, multi-scale testing, and model ensemble. "800×" means that the short side of input images is scaled to 800.

| Method	Venue	Backbone	IC19-ReCTS Test Set			
			P	R	F	1-NED
Tencent-DPPR Team* [32]	Competition	-	93.49	92.49	92.99	81.45
SANHL_v1* [32]	Competition	-	91.98	93.86	92.91	81.43
HUST_VLRGROUP* [32]	Competition	-	91.87	92.36	92.12	79.38
EAST [33]	CVPR'17	VGG16	74.25	73.65	73.95	-
PSENet [29]	CVPR'19	ResNet50	87.31	83.86	85.55	-
FOTS [19]	CVPR'18	ResNet50	78.25	82.49	80.31	50.83
Mask TextSpotter [13]	TPAMI	ResNet50	89.30	88.77	89.04	67.79
Detection Only	-	ResNet50	90.72	90.73	90.72	-
AE TextSpotter w/o LM	-	ResNet50	91.54	90.78	91.16	70.95
AE TextSpotter (800×)	-	ResNet50	92.60	91.01	91.80	71.81
AE TextSpotter (1000×)	-	ResNet50	**92.28**	**91.45**	**91.86**	**73.12**

4.6   Discussion

As demonstrated in previous experiments, the proposed AE TextSpotter works well in most cases, including scenarios of text detection ambiguity. Nevertheless, there is still room for improvement in some parts as follows. **1)** The recognition performance of our method is not high (only 51.32% in Table 4), supressing the effectiveness of LM. **2)** TDA-ReCTS is selected by rules that only give a vague description of ambiguity in text detection. Therefore, a manually selected dataset is important for further exploring text detection ambiguity. **3)** The proposed AE TextSpotter is a baseline for removing text detection ambiguity, and it is still worthwhile to further explore a better way to combine visual and linguistic representation.

Although our AE TextSpotter works well in most cases. It still fails in some intricate images, such as text-like regions, strange character arrangement, and

Table 6. The time cost of all modules in AE TextSpotter.

Scale	FPS	Time Cost per Image (ms)					
		Backbone	RPN	TDM	CRM	LM	Post-proc
800×	1.3	31.3	35.1	440.0	122.3	118.6	3.8
1000×	1.2	63.1	42.6	487.2	127.8	121.1	3.9

(a) Large Character Spacing   (b) Juxtaposed Text Lines
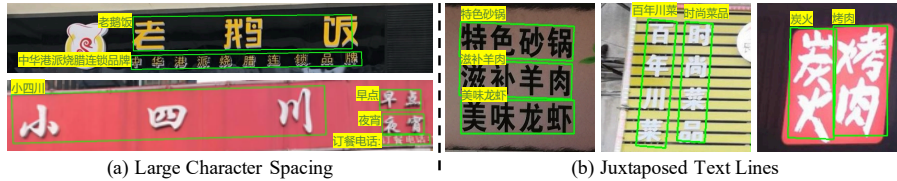
Fig. 8. Visualization results of AE TextSpotter on TDA-ReCTS.

texts in fanc styles. More details about these failed cases can be found in Appendix A.2.

5   Conclusion & Future Work

In this paper, we first conducted a detailed analysis of the ambiguity problem in text detection, and revealed the importance of linguistic representation in solving this problem. Then we proposed a novel text spotter, termed AE TextSpotter, which introduces linguistic representation to eliminate ambiguity in text detection. For the first time, linguistic representation is utilized in scene text detection to deal with the problem of text detection ambiguity. Concretely, we propose the Language Module (LM), which can learn to re-score the text proposals by linguistic representation. LM can effectively lower the scores of incorrect text lines while improve the scores of correct proposals. Furthermore, a new validation set is proposed for benchmarking the ambiguity problem. Extensive experiments demonstrate the advantages of our method, especially in scenarios of text detection ambiguity.

In the future, we will strive to build a more efficient and fully end-to-end network for ambiguous text spotting, and plan to build a large-scale and more challenging dataset which is full of ambiguous text samples.

Acknowledgments

References

1. Cheng, Z., Bai, F., Xu, Y., Zheng, G., Pu, S., Zhou, S.: Focusing attention: Towards accurate text recognition in natural images. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5076–5084 (2017)
2. Cheng, Z., Xu, Y., Bai, F., Niu, Y., Pu, S., Zhou, S.: Aon: Towards arbitrarily-oriented text recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5571–5579 (2018)
3. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
4. Deng, D., Liu, H., Li, X., Cai, D.: Pixellink: Detecting scene text via instance segmentation. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
7. Feng, W., He, W., Yin, F., Zhang, X.Y., Liu, C.L.: Textdragon: An end-to-end framework for arbitrary shaped text spotting. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9076–9085 (2019)
8. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd international conference on Machine learning. pp. 369–376. ACM (2006)
9. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
11. Li, H., Wang, P., Shen, C.: Towards end-to-end text spotting with convolutional recurrent neural networks. arXiv preprint arXiv:1707.03985 (2017)
12. Li, X., Wang, W., Hou, W., Liu, R.Z., Lu, T., Yang, J.: Shape robust text detection with progressive scale expansion network. arXiv preprint arXiv:1806.02559 (2018)
13. Liao, M., Lyu, P., He, M., Yao, C., Wu, W., Bai, X.: Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. IEEE transactions on pattern analysis and machine intelligence (2019)
14. Liao, M., Shi, B., Bai, X.: Textboxes++: A single-shot oriented scene text detector. IEEE transactions on image processing **27**(8), 3676–3690 (2018)
15. Liao, M., Shi, B., Bai, X., Wang, X., Liu, W.: Textboxes: A fast text detector with a single deep neural network. In: AAAI. pp. 4161–4167 (2017)
16. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117–2125 (2017)
17. Liu, J., Liu, X., Sheng, J., Liang, D., Li, X., Liu, Q.: Pyramid mask text detector. arXiv preprint arXiv:1903.11800 (2019)
18. Liu, W., Chen, C., Wong, K.Y.K., Su, Z., Han, J.: Star-net: A spatial attention residue network for scene text recognition. In: BMVC. vol. 2, p. 7 (2016)

19. Liu, X., Liang, D., Yan, S., Chen, D., Qiao, Y., Yan, J.: Fots: Fast oriented text spotting with a unified network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5676–5685 (2018)
20. Long, S., Ruan, J., Zhang, W., He, X., Wu, W., Yao, C.: Textsnake: A flexible representation for detecting text of arbitrary shapes. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 20–36 (2018)
21. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems. pp. 8024–8035 (2019)
22. Qin, S., Bissacco, A., Raptis, M., Fujii, Y., Xiao, Y.: Towards unconstrained end-to-end text spotting. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4704–4714 (2019)
23. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
24. Shi, B., Bai, X., Belongie, S.: Detecting oriented text in natural images by linking segments. arXiv preprint arXiv:1703.06520 (2017)
25. Shi, B., Bai, X., Yao, C.: An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE transactions on pattern analysis and machine intelligence **39**(11), 2298–2304 (2016)
26. Shi, B., Yang, M., Wang, X., Lyu, P., Yao, C., Bai, X.: Aster: An attentional scene text recognizer with flexible rectification. IEEE transactions on pattern analysis and machine intelligence (2018)
27. Tian, Z., Huang, W., He, T., He, P., Qiao, Y.: Detecting text in natural image with connectionist text proposal network. In: European Conference on Computer Vision. pp. 56–72. Springer (2016)
28. Wang, J., Hu, X.: Gated recurrent convolution neural network for ocr. In: Advances in Neural Information Processing Systems. pp. 335–344 (2017)
29. Wang, W., Xie, E., Li, X., Hou, W., Lu, T., Yu, G., Shao, S.: Shape robust text detection with progressive scale expansion network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9336–9345 (2019)
30. Wang, W., Xie, E., Song, X., Zang, Y., Wang, W., Lu, T., Yu, G., Shen, C.: Efficient and accurate arbitrary-shaped text detection with pixel aggregation network. In: Proceedings of the IEEE International Conference on Computer Vision (2019)
31. Xie, E., Zang, Y., Shao, S., Yu, G., Yao, C., Li, G.: Scene text detection with supervised pyramid context network. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 9038–9045 (2019)
32. Zhang, R., Zhou, Y., Jiang, Q., Song, Q., Li, N., Zhou, K., Wang, L., Wang, D., Liao, M., Yang, M., et al.: Icdar 2019 robust reading challenge on reading chinese text on signboard. In: 2019 International Conference on Document Analysis and Recognition (ICDAR). pp. 1577–1581. IEEE (2019)
33. Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., Liang, J.: East: An efficient and accurate scene text detector. arXiv preprint arXiv:1704.03155 (2017)