A. The OCR-CC Dataset

Figure A. (a,b) The distribution of the detected scene text number by Microsoft-OCR on the Conceptual Captioning (CC) dataset \cite{Gao2017Conceptual} and our OCR-CC dataset. (c,d) Representative examples of discarded and selected images. We draw the OCR box over multiple related words for visualization purposes. We note that each scene text region contains a single word, e.g., four words “HYUNDAI,” “INSPIRING,” “THE,” “FL” in the top left sub-figure of (d).

In this section, we introduce the details of building the OCR-CC dataset based on the Conceptual Captioning (CC) dataset \cite{Gao2017Conceptual}. First, we run the Microsoft Azure OCR system on all CC images (around 3.1 million). Then, we discard the images that don’t have scene text (around half of the CC images) or have watermark “text” only (around 5% of the CC images). These watermark “text” records the source image website/provider and are thus not related to the image content. Figure A (c) shows examples of the discarded images, which either have no detected scene text or have watermark “text” only. In the end, we select 1,367,170 images from CC as the images in our OCR-CC dataset. We pair each selected image with a caption \( w \) for pre-training. The caption text \( w \) is the concatenation of the original image caption \( w^d \) in CC, the detected object labels \( w^{\text{obj}} \), and the detected scene text words \( w^{\text{ocr}} \). Figures A (a,b) visualize the distribution of the scene text number in CC and our OCR-CC, respectively. Similar to the distribution on TextVQA \cite{Oh2018TextVQA} and ST-VQA \cite{Nguyen2019ST}, the majority of images contain 3-10 detected scene text regions, while a small portion of images has a large number of scene text regions. Figure A (d) shows some representative selected images.

B. TextCaps Results

Tables A, B present the full results on TextCaps \cite{Yao2018TextCaps} to supplement the abstracted results in the main paper’s Table 3. We draw similar conclusions from Tables A, B as the ones in the main paper. Specifically, “TAP” significantly improves the non-TAP baseline “M4C†” in all metrics with the identical network architecture and training data. Our TAP approach also outperforms the previous state of the art \cite{Yao2018TextCaps, Nguyen2019ST, Zhao2019Cross} by large margins.

Furthermore, we compare TAP with the oracle numbers, as shown in the gray text color at the bottom part of Table A.
bles A, B. “TAP” outperforms the “M4C (GT OCR)” that uses ground-truth scene text detection in training and inference. Meanwhile, there still exists a gap between “TAP” and human performance. We expect future studies focusing on captioning to further reduce the gap, e.g., with better decoding step pre-training designed especially for captioning.

C. Hyper-parameters

We summarize the hyper-parameters used in the “TAP” and “TAP††” experiments. We conduct experiments based on the M4C [3, 6] and follow most of its hyper-parameter selections, as shown in Table C. We highlight the changed parameters in bold in the table.

- First, the max length of the extended text input \(w = [w^q, w^{obj}, w^{ocr}]\) is set to \(20 + 100 + 100 = 220\).
- Second, we increase the max length of scene text \(v^{ocr}\) from 50 to 100 when experimented with Microsoft-OCR. Compared with Rosetta, Microsoft-OCR generates more detected scene text regions in each image. For example, in the TextVQA dataset, the mean and median of scene text numbers are 12.8 and 8 with Rosetta, and are 23.1 and 12 with Microsoft-OCR. With Rosetta, 3.5% of images contain more than 50 scene text regions detected, while the percentage is 14.3% with Microsoft-OCR. To cover more detected scene text, we increase the max length of scene text \(v^{ocr}\) from 50 to 100 when experimented with Microsoft-OCR.
- In the experiment of “pre-training without extra data” (“TAP††”), we follow the same learning rate step and maximum iteration settings as used in the fine-tuning. In pre-training with OCR-CC (“TAP†††”), we pre-train the model for a maximum iteration of \(480K\) and scale the learning rate steps linearly.

D. Pre-train + Fine-tune vs. Joint-train

Results in the main paper’s Section 4.3 show that TAP works well even without extra data. We hypothesize that we can view TAP as a multi-task learning framework, and obtain similar improvement by using the pre-training tasks (MLM, ITM, RPP) as the auxiliary training loss. Therefore, we explore an alternative training pipeline named “joint train,” where the pre-training tasks are used as the auxiliary losses together with the main answer/caption loss. Because MLM and ITM tasks require “polluting” the input sequence, we randomly select 50% of the samples in a batch to compute the pre-training loss and keep the remaining 50% unchanged for the answer/caption loss.

Studies show that these two training pipelines can achieve similar performances, i.e., 49.91% for “pre-train + fine-tune” and 49.46% for “joint train” on TextVQA.

E. Qualitative Results

In this section, we present additional qualitative examples. Figure B shows the failure cases that can be corrected by OCR detection. Figure C presents the failure cases of our method. “TAP” occasionally fails on samples that require complex reasoning (Figures C (a,b)) or have incorrect scene text detection (Figures C (c,d)). For example, in Fig-
Figure B. Failure cases that can be corrected by scene text detection. The top and bottom rows visualize the detected scene text by Rosetta-OCR and Microsoft-OCR, respectively. We draw adjacent words into the same box for visualization purposes and highlight the key scene text regions for the question, e.g., “moon bar,” “bud light,” “clemson,” and “marvel.”

Figure C. Representative failure cases of “TAP.” We highlight the key scene text regions for each question.

ure C (a), TAP selects the scene text “cutfittep” on the black bag as the answer, instead of the correct scene text “aldo” on the referred white bag.

References


