Indian Institute of Technology Hyderabad
Computer Science & Engineering Department

Amazon Machine Learning Challenge, 2024

Submitted By -

Harikrishnan V

Aishwarya Dash

Akkinepally Phani Sahasra

Suryansh Gautam

1. PROBLEM STATEMENT

The problem for the ML hackathon involves developing a machine learning model that extracts specific entity values, like weight, volume, and dimensions, from product images. The dataset includes image links, product categories, entity names (e.g., item weight, height), and corresponding entity values.

Key Points:

Goal: Extract and predict entity values from images in the format x unit (e.g., "2 gram").

Data: Training data contains image links, product category, entity names, and entity values. Test data omits the entity values.

Output: A CSV file with index and prediction columns, formatted exactly as shown in a provided sample output file.

Units: Predictions must use specific allowed units (e.g., grams, centimetres) based on the entity type.

Evaluation: Predictions are evaluated using F1 score, based on how well the model predicts entity values compared to the ground truth.

2. METHODOLOGY

In this hackathon, the objective was to extract entity values like weight, volume, and dimensions from product images using a combination of Optical Character Recognition (OCR) and Natural Language Processing (NLP) models. The approach involved using PaddleOCR for extracting text from images, followed by a post-processing and tokenization pipeline. The final step utilized a T5 model for predicting the correct entity values by combining entity names and the extracted text in a prompt-based format.

Models Tested:

Image Processing Tasks :

1. Tesseract OCR
2. Easy OCR and
3. PaddleOCR.

Entity Extraction from Gathered Text Data :

1. **T5-small:** Baseline model for sequence-to-sequence tasks.
2. **BART-base**: Used for sequence generation and comparing model architectures.
3. **DistilBART**: A lighter version of BART for faster inference.

The detailed steps of implementation can be found below :

1. Data Preparation and Image Extraction
The dataset provided contains product images with corresponding entity_name (like "item_weight", "height") and, for the training set, entity_value.

Image Link Retrieval: We used the download_images function provided in src/utils.py to download product images. These images served as the input for PaddleOCR.

2. PaddleOCR for Text Extraction

After experimenting on PaddleOCR, EasyOCR and Tesseract OCR, we found that PAddleOCR provided the most accurate text data extraction.

OCR Extraction: We used PaddleOCR to extract text from the product images. PaddleOCR is an open-source deep learning-based tool for text detection and recognition from images. It can identify text regions, extract the text, and provide bounding boxes for the recognized text regions.

OCR Output: For each image, PaddleOCR returned:

1. Extracted text from the image.
2. Bounding boxes corresponding to the location of each text block.
3. Confidence scores for the extracted text regions.

Storing OCR Results: The extracted OCR results were stored in a DataFrame that included:

image: The image identifier.
concatenated_text: A string combining all the extracted text from the image.
concatenated_boxes: The coordinates of the bounding boxes.
concatenated_confidences: The confidence scores associated with each piece of text.
entity_name: The entity name (e.g., "item_weight").
entity_value: The target entity value for the training set (this was absent in the test set).

3. Post-processing and Tokenization
After extracting the text using PaddleOCR, post-processing was necessary to clean and prepare the data for the T5 model:

Text Cleaning:

Removed irrelevant characters or symbols (such as special characters that are not part of the entity value).
Normalized text by converting units or synonyms to a standard format (e.g., "gm" to "gram").
Tokenization:

Split the concatenated text into tokens, representing individual words, numbers, or unit terms. This helped in pairing specific parts of the text with the corresponding entity names.

4. Formatting for T5 Model

The T5 model (Text-to-Text Transfer Transformer) was employed for the final prediction task. Since T5 is a sequence-to-sequence model, the data was formatted as prompts for input to the model.

Prompt-Based Question Answering: For each sample, we created a prompt by concatenating the entity_name with the concatenated_text from the OCR extraction. The format used was:

entity_name ? concatenated_text

For example, if the entity_name was "item_weight" and the extracted text was "Garnier Face Cream Net weight 34 gm, 56 ml ", the prompt would be:

item_weight ? Garnier Face Cream Net weight 34 gm, 56 ml

This prompt-based input encouraged the model to focus on the specific entity and locate its associated value within the text.

5. Training the T5 Model

Objective: The T5 model was tasked with predicting the correct entity_value from the input prompt.

Data Split: We trained the model using the training data, which included both the entity_name and entity_value. The model was fine-tuned on this task to predict the correct value (e.g., "34 gram") based on the combined prompt.

Loss Function: The T5 model was trained using a cross-entropy loss, typical for sequence prediction tasks. The model learned to maximize the probability of generating the correct entity value based on the given input prompt.

6. Prediction on Test Data
For the test data, where the entity_value was not provided, the same approach was followed:

Input: The test images were processed using PaddleOCR, the entity_name was concatenated with the extracted text, and the resulting prompt was passed to the T5 model.

Output: The model generated predictions in the required format: "x unit" (e.g., "34 gram"). If no valid value was predicted, we returned an empty string "".

7. Post-processing of Predictions

After the model predicted the values, we ensured that the predictions followed the required format:

Unit Validation: Predictions were checked to ensure they used one of the allowed units specified in the entity_unit_map. If an invalid unit was detected, the prediction was corrected or discarded (resulting in an empty string).

Formatting Consistency: The output predictions were post-processed to ensure they were formatted as required (e.g., "2 gram" instead of "2 gms" or "2.2e2 kilogram"). This was crucial for passing the sanity checker provided in the challenge.

8. Final Output
The final output was a CSV file with two columns:

index: The unique identifier of each test sample.
prediction: The predicted entity value in the format "x unit", or an empty string "" if no value was found.
This file was passed through the sanity checker (src/sanity.py) to ensure it met the required formatting rules before submission.

3. EXPERIMENTAL ANALYSIS

Batching: Dataset was divided into batches to optimize memory usage during training.

GPU P100: Training and experiments were conducted using an NVIDIA Tesla P100 for faster processing.

Models Tested:

Image Processing Tasks :

4. Tesseract OCR
5. Easy OCR and
6. PaddleOCR.

Entity Extraction from Gathered Text Data :

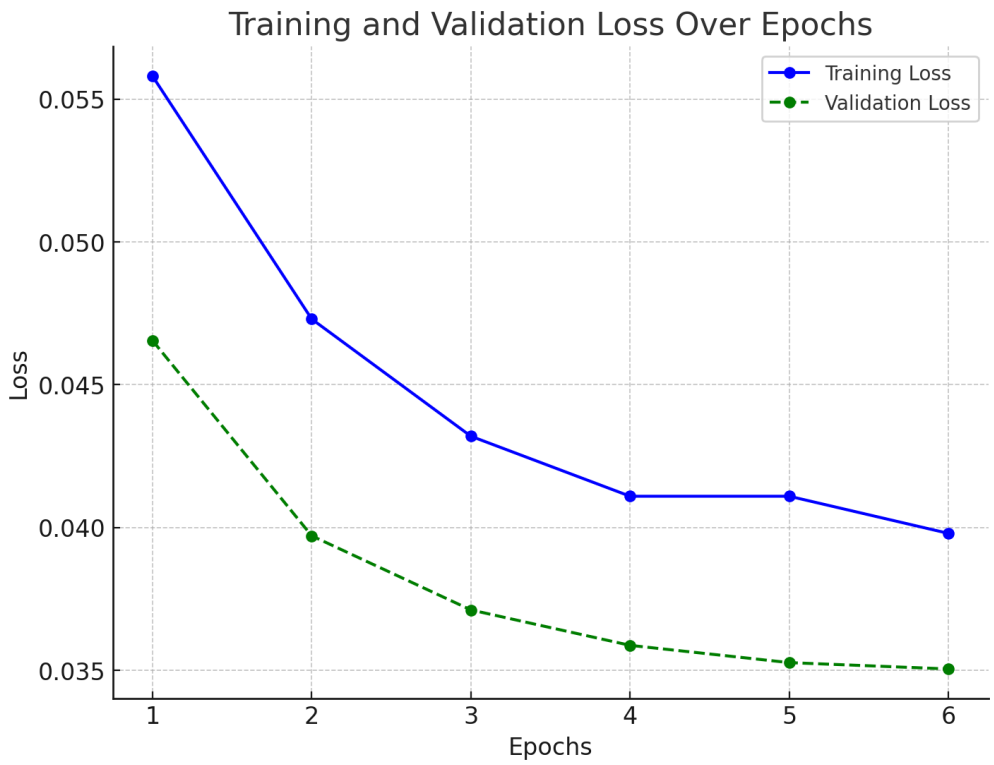
4. **T5-small:** Baseline model for sequence-to-sequence tasks.
5. **BART-base**: Used for sequence generation and comparing model architectures.
6. **DistilBART**: A lighter version of BART for faster inference.

Training: Models trained using concatenated prompts of entity_name and extracted text.

Evaluation: Models were evaluated based on F1 score to assess precision and recall on test data.

The following table shows a glimpse of the training versus validation loss :

Epoch	Training Loss	Validation Loss
1	0.055800	0.046547
2	0.047300	0.039715
3	0.043200	0.037122
4	0.041100	0.035882
5	0.041100	0.035278
6	0.039800	0.035056



4. CONCLUSION

In this project, we developed a robust pipeline for extracting entity values from product images by combining PaddleOCR for text extraction and used the concept of question-answering task by using transformer models like T5-small for prediction. By leveraging batch processing and the computational power of a GPU P100, we efficiently trained and tested multiple models. The results demonstrated that each model performed well in its respective task, with a focus on precision and recall measured by the F1 score. This approach offers an effective solution for extracting key information from images in domains like e-commerce and healthcare.