Report – Team 'Briefcase'

AMAZON ML CHALLENGE 2024

Shantanu Yadav
Department of Biotechnology, IIT Kharagpur
shantanu19@kgpian.iitkgp.ac.in

Yash Lanjewar
Department of Architecture & Regional Planning, IIT Kharagpur
yashlanjewar597@kgpian.iitkgp.ac.in

Indrajit Chaudhuri
Department of Mechanical Engineering, IIT Kharagpur
indrajit.chaudhuri@kgpian.iitkgp.ac.in

Through this document, we describe concisely the procedure followed by us with regard to processing the data, designing and using model pipelines, and obtaining results.

DATASET

The data, given in the form of a '.csv' file, contains 4 fields, namely image link, group-ID, entity name, and entity value. There are 263,859 records in the training set (131,187 in testing set), which are predominantly images of common objects and household commodities, with a small fraction of erroneous entries (black or unclear, etc. images). Furthermore, there is a relatively small number of unique group-ID values, indicating that the number of distinct objects under consideration is approximately only around 750.

MACHINE LEARNING TECHNIQUES EMPLOYED

The first approach we devised was as follows:

Approach I

i. Start by using an Optical Character Recognition (OCR) technology to obtain individual characters and text within each of the images, after preprocessing steps such as binarization, noise removal, resizing and resolution improvement, contrast enhancement, etc
ii. Thereafter, train a model such as a transformer to identify exactly which value and which unit should be considered as the final output, conforming to the rules of submission

Easy_OCR, Paddle_OCR, Tesseract were three separate modules that we experimented with, all of which provided somewhat decent results to varying degrees of accuracy (Paddle_OCR being the best out of all we tried). However, there seemed to be a major drawback to this approach, since not all images contained text in a format readable by OCR. Moreover, certain images did not have any additional supporting clues as to the exact text that needed to be extracted, thereby increasing the task difficulty for the

transformers. Upon receiving unsatisfactory results, we proceeded to undertake a different strategy, as outlined next.

Approach II

i. Experiment with Vision-supported Large Language models to identify the best-suited ones for the task at hand
ii. Fine-tune the best model according to our dataset, pre-process the data as required by this specific model, and evaluate the results

LLaVA and Qwen2 were the 2 open-source models we picked. LLaVA represents a novel end-to-end trained large multimodal model that combines a vision encoder and Vicuna for general-purpose visual and language understanding, achieving impressive chat capabilities mimicking spirits of the multimodal GPT-4. We fine-tuned the LLaVA-1.6 model on a subset of the given training dataset; however, owing to our limited computational resources, the minimal number of epochs that could be executed did not yield sufficiently good results. This somewhat decent set of predictions constituted our first submission. Next, we experimented with Qwen2 in a similar fashion as we did with LLaVA. Qwen2 is a language model series including decoder language models of different model sizes, each of which comprises a base language model and the aligned chat model. It is based on the Transformer architecture with SwiGLU activation, attention QKV bias, group query attention, mixture of sliding window attention and full attention, etc. as well as an improved tokenizer adaptive to multiple natural languages and codes.

Qwen2 seemed to be the better-performing algorithm, and resulted in our highest-scoring submission.

PROCESSING OF MODEL OUTPUT

The outputs produced by the models naturally contained several inconsistencies and hence, did not fulfil the criteria for acceptable submissions. Therefore, we designed a separate script in Python to clean the predictions, and modify them wherever necessary, to ensure compatibility with the specified output guidelines. This script extracted the value of the entity, identified locations where a unit could not be detected and converted these to blanks, and also changed the units exactly to the required formats.