Laboratory file

on

Agentic AI

School of Engineering and Technology

Department of Computer Science and Engineering

Subject code – **CSCR3215**

SUBMITTED BY:

Name : Syed Maaz Ahmad
Roll No: 2301010893

SUBMITTED TO:

Mr. Ayush Singh

Sharda University

Greater Noida, Uttar Pradesh

BLIP (Bootstrapping Language-Image Pre-training) model

1. Setup and Installation

The notebook begins by installing the necessary libraries: evaluate, transformers, datasets, and accelerate. These are essential for loading the dataset, managing the model, and handling the training process.

2. Dataset Loading and Preprocessing

- Dataset: The project uses the "conceptual_captions" dataset, a large collection of image-URL and caption pairs.
- Loading: The datasets library is used to load a small portion of the dataset for demonstration purposes (100 training and 10 validation samples).
- Processor: A BlipProcessor is loaded from the pre-trained "Salesforce/blip-image-captioning-base" model. This processor handles both image transformations (resizing, normalization) and text tokenization.
- Preprocessing Function: A function named preprocess is defined to prepare the data for the model. It takes a batch of examples, processes the images with the BlipImageProcessor, and tokenizes the corresponding captions with the BlipTokenizer. The tokenized captions are set as the labels for the model.
- Applying Preprocessing: The preprocess function is applied to the training and validation datasets using the map function, with batching enabled for efficiency.

3. Model Definition

- Model: A BlipForConditionalGeneration model is loaded from the pre-trained "Salesforce/blip-image-captioning-base" checkpoint. This model is designed specifically for tasks where text is generated based on an image input.

4. Training

- Evaluation Metric: The ROUGE score is chosen as the evaluation metric, which is standard for text generation tasks. It is loaded using the evaluate library.
- Training Arguments: Seq2SeqTrainingArguments are configured for the training process. Key parameters include:
    - output_dir: Directory to save the trained model.
    - learning_rate: The learning rate for the optimizer.
    - per_device_train_batch_size and per_device_eval_batch_size: Batch sizes for training and evaluation.
    - num_train_epochs: The number of training epochs.
    - weight_decay: Weight decay for regularization.
    - evaluation_strategy: Evaluation is set to be performed after each epoch.
    - save_strategy: The model is saved after each epoch.

    - predict_with_generate: Enabled to generate captions during the evaluation phase.
- Collation: A default_data_collator is used to batch the preprocessed data into tensors.
- Evaluation Function: A compute_metrics function is defined to calculate the ROUGE score during evaluation. It decodes the generated predictions and the ground-truth labels, then computes the score.
- Trainer: A Seq2SeqTrainer is instantiated with the model, training arguments, data collator, datasets, and the compute_metrics function.
- Training Execution: The train() method of the trainer is called to start the fine-tuning process.

5. Inference and Evaluation

- Loading the Fine-Tuned Model: The fine-tuned model is loaded from the saved checkpoint.
- Inference on a Sample Image:
    - An example image from the validation set is chosen.
    - The image is preprocessed using the BlipImageProcessor.
    - The generate method of the model is used to create a caption for the image.
    - The generated caption is decoded and printed, demonstrating the model's ability to describe the image content.
- Pushing to Hub (Optional): The notebook includes code to push the fine-tuned model and the processor to the Hugging Face Hub, allowing for easy sharing and reuse. (This step requires logging into a Hugging Face account).