Individual Project Report
Daqian Dang
DATS 6312
May 6, 2023

Text Summarization of News Articles for Daily/Topic News App

I. An Overview of the Project

Our project aimed to create an app that quickly summarizes news articles and reports, helping decision-makers stay updated. We used the T5-small model, a powerful tool for text summarization, to achieve this goal.

We utilized a dataset from Kaggle, titled "CNN-Daily Mail News Text Summarization." The dataset contains full-text news articles from CNN and Daily Mail, along with summarized highlights written by the article authors. The dataset is structured with full text as input and summarized highlights as the target. It consists of 287,113 training samples, 13,368 validation samples, and 11,590 test samples. On average, articles contain 781 tokens, while summaries consist of 56 tokens.

The T5-small model was employed, which is an advanced language processing model that can capture complex patterns in text and generate high-quality summaries. We chose this model for its excellent performance in our specific project – text summarization.

To measure our model's performance, we used the ROUGE metric. ROUGE is a popular method for evaluating text summarization quality by comparing the model's output to human-written summaries. This metric helped us assess how well our T5-small based model performed.

II. Description of Individual Work

Initially, I focused on developing the preprocessing code, configuring the T5-small model, and laying the groundwork for the training parameters. Additionally, I designed the compute_metrics function and established the Seq2SeqTrainingArguments and Seq2SeqTrainer, using resources from the Hugging Face website as a reference.

Andrew took charge of reviewing the initial codebase, identifying and correcting errors, and pinpointing areas that required improvement. His contributions ensured that the project would function seamlessly within a Python environment.

Rajkumar played a crucial role in refining the model's performance by adjusting the fine-tuning hyperparameters and overseeing the training process. His efforts led to the optimization of the model, yielding the best possible results.

III. Describe the Portion of My Work on the Project

Preparing the dataset: The code defines a custom PyTorch dataset class called TextSummarizationDataset, which takes a dataframe, tokenizer, and maximum input and output lengths. The class tokenizes the source text and target summaries using the provided tokenizer and pads or truncates them to the specified maximum lengths. It then returns the input_ids and attention masks for the source text and labels (target summaries) for each example.

Creating DataLoaders: The code creates instances of the custom dataset for both the training and testing sets. It then constructs PyTorch DataLoaders for these datasets, which enable efficient loading and batching of the data during training and evaluation.

Setting up the model and training parameters: After conducting research on the T5 model, as described in the relevant research paper, it was determined that this model would be a suitable choice for the project as this is a text summarization task. Consequently, the code initializes the tokenizer and T5-small model using the specified checkpoint. It then defines the training parameters, including learning rate, weight decay, batch size, and gradient accumulation steps, which were chosen to optimize the model's performance for this specific task. Below are my initial training parameters example:

CHECKPOINT	t5-small
LEARNING_RATE	1e-4
WEIGHT_DECAY	1e-2
N_EPOCH	1
BATCH_SIZE	16
PREFETCH_FACTOR	2
N_WORKER	4
GRADIENT_ACCUMULATION_STEPS	2
MAX_INPUT_LEN	512
MAX_OUTPUT_LEN	128

Defining the metric and evaluation function: The code defines a function called compute_metrics, which takes the model's predictions and labels, decodes them, and computes the ROUGE metric using the evaluate package. Initially, the code utilized the deprecated dataset.load_metric function, which led to unsuccessful execution. To resolve this issue and ensure smooth and accurate operation, the deprecated function was replaced with the evaluate package, which effectively solved the problem.

Training the model: The code sets up a Seq2SeqTrainer with the model, training arguments, dataset, tokenizer, data collator, and the compute_metrics function. The trainer is used to train the model with the specified parameters.

IV. Results

Based on my initial training parameters, the T5-small model was trained on the dataset containing 249,576 examples with one epoch and an effective batch size of 32. The training process took approximately 5 hours, resulting in a training loss of 1.175, indicating the model's ability to generate summaries from news articles. Please see training table indicated below.

Running Training Table	
Number of examples	249576
Number of Epochs	1
Batch size per device	16
Total train batch size (with parallel, distributed & accumulation)	32
Gradient Accumulation steps	2
Total optimization steps	7799
Number of trainable parameters	60506624
Train_runtime	21471.465
Train_loss	1.175

The evaluation results of the T5-small model for the text summarization task provide valuable information about the model's performance. The ROUGE metric was used to assess the quality of the generated summaries by comparing them with human-written reference summaries. Please see evaluation table showed below.

Running Evaluation Table	
Number of examples	62395
Number of epochs	1
Batch size	16
Eval_loss	1.0422
Eval_rouge1	0.2378
Eval_rouge2	0.1083
Eval_rougeL	0.197
Eval_rougeLsum	0.197
Eval_gen_len	18.983
Eval_runtime	3647.765
Eval_samples_per_second	17.105
Eval_steps_per_second	1.069

Eval_loss: 1.0422 - This value represents the model's evaluation loss, which measures the discrepancy between the model's predictions and the target summaries. A lower value indicates better performance.

Eval_rouge1: 0.2378 - This is the ROUGE-1 score, which measures the overlap of unigrams (single words) between the generated summaries and the reference summaries. The score ranges from 0 to 1, with a higher value indicating better performance.

Eval_rouge2: 0.1083 - This is the ROUGE-2 score, which measures the overlap of bigrams (two-word sequences) between the generated summaries and the reference summaries. A higher value signifies better performance.

Eval_rougeL: 0.197 - This is the ROUGE-L score, which measures the longest common subsequence between the generated summaries and the reference summaries. A higher value represents better performance.

Eval_rougeLsum: 0.197 - This score is the same as ROUGE-L, as it also measures the longest common subsequence.

Eval_gen_len: 18.983 - This value represents the average length of the generated summaries in terms of tokens. It provides insight into the length and conciseness of the model's output.

Eval_runtime: 3647.765 - This value denotes the total runtime of the evaluation process, measured in seconds. The evaluation process took approximately about 1 hour to complete.

V. Summary and Conclusion

The training and evaluation results indicate that the model's performance in generating summaries can be improved, as the ROUGE scores are relatively low. Enhancing the model's summarization capabilities would require exploring various optimization strategies, such as fine-tuning the hyperparameters, increasing the number of training epochs, experimenting with different model sizes, and performing more advanced text preprocessing techniques.

It is important to note that the training process was computationally expensive, taking almost 6 hours to complete. This highlights the need for efficient and cost-effective training techniques, especially when working with limited resources. Additionally, exploring the use of more powerful hardware or distributed training methods could help reduce the training time and make the process more manageable.

While the current model has achieved a reasonable level of summarization performance, further experimentation and optimization are necessary to improve its capabilities. Balancing the need for better performance with computational efficiency and resource constraints remains a critical challenge in developing a robust text summarization model.

VI. Code Breakdown

Total lines contributed: 78
Number of copied lines: 30
Number of lines modified: 6
Number of my work lines: 48
Score = (30-6)/(30+48)*100 = 30.8

References:

CNN-DailyMail News Text Summarization. (n.d.). CNN-DailyMail News Text Summarization | Kaggle. https://www.kaggle.com/datasets/gowrishankarp/newspaper-text-summarization-cnn-dailymail

cnn_dailymail · Datasets at Hugging Face. (2023, January 24). Cnn_Dailymail · Datasets at Hugging Face. https://huggingface.co/datasets/cnn_dailymail

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. arXiv preprint arXiv:1910.10683. https://doi.org/10.48550/arXiv.1910.10683

Alammar, J. (2018, June 27). *The Illustrated Transformer*. The Illustrated Transformer – Jay Alammar – Visualizing Machine Learning One Concept at a Time. http://jalammar.github.io/illustrated-transformer/