Final Solution Report on Text De-toxification
Practical Machine Learning and Deep Learning - Assignment 1

Jayveersinh Raj,
BS20-DS01,
j.raj@innoplis.university

1. Introduction

Prompting adds some task-specific input text to assist direct language model behavior. The process of just training and updating freshly added prompt tokens to a pretrained model is known as prompt tuning. Instead of completely fine-tuning a distinct model, you can use a single pretrained model with frozen weights and train and update a smaller number of prompt parameters for each downstream task. Prompt adjustment can be more effective as models get bigger, and outcomes improve even more when model parameters increase [1]. Additionally by injecting trainable rank decomposition matrices into each layer of the Transformer architecture and freezing the pre-trained model weights, low-Rank Adaptation (LoRA) or Quantized Low-Rank Adaptation (QLoRA) significantly reduce the number of trainable parameters for downstream tasks [2].

2. Data description

HuggingFace datasets library was used to load the data that was prepared from raw data, which was discussed in the Solution building report. However, only 99,000 samples were taken for training, the rest 1000 were kept aside for testing. These are 9000 more training samples than t5-small training set. This 9000 more were given to the model because it would only be prompt tuned for around 0.13 epochs. The test set remains the same as was for baseline solution, and hypothesis 1 solution (seq2seqLM) for objective deduction of performance. Datasets library was used for both hypothesis 1 (seq2seqLM) and the current final solution mainly because it processes massive datasets with zero-copy reads without any memory limits for maximum speed and efficiency, supported by the Apache Arrow format[3].

2.1. *Prompts creation*

2.1.1. *For training*

The dataset as described above was then mapped into a prompt format as it could be seen in the picture below:

```
def create_prompt(toxic, detoxified):
  if len(detoxified) < 1:
    detoxified = "Cannot Find Answer"
  else:
    detoxified = detoxified
  prompt_template = f"### Detoxify:\n{toxic}\n\n### detoxified:\n{detoxified}</s>"
  return prompt_template

mapped_data_train = data["train"].map(lambda samples: tokenizer(create_prompt(samples['toxic'], samples['detoxified'])))
mapped_data_test = data["test"].map(lambda samples: tokenizer(create_prompt(samples['toxic'], samples['detoxified'])))
```

This is ensure the model understands the tasks, and to have a clear visualization of the task.

2.2. *For inference*

However, for inference since, we want the model to generate the detoxified verison, the prompt was the same except after the word **detoxification** as can be seen below:

```
def make_inference(toxic):
  batch = tokenizer(f"### Detoxify:\n{toxic}\n\n### detoxified:\n", return_tensors='pt')

  with torch.cuda.amp.autocast():
    output_tokens = qa_model.generate(**batch, max_new_tokens=200)

  return tokenizer.decode(output_tokens[0], skip_special_tokens=True).split("\n")[-1]
```

3. Model description

Bloom-3b which is a decoder-style Transformer-based language model developed by BigScience was used for prompt tuning. Due to computational limitations, 3 billion (8-bit quantize) is possibly the largest model that could be loaded on a free Google Colab T4 GPU (15GB). According to [4] by using low-precision data types, such as 8-bit integers (int8) rather than the more common 32-bit floating point data types (float32), to represent the weights and activations, quantization is a technique that can lower the computational and memory costs associated with executing inference.

In theory, a model with fewer bits uses less energy and memory storage, and operations like matrix multiplication can be completed considerably more quickly using integer arithmetic. Additionally, it enables the use of models on embedded systems, which occasionally can only handle integer data types. Nevertheless, the choice of such a model to bloom is because of its nature of open source, and multilingual capabilities. Below are the technical details of the model:

- **Architecture**: Modified from Megatron-LM GPT2 (decoder only architecture)
- **Parameters**: 3,002,557,440
- **Sequence length**: 2048 tokens (model can see up to previous 2048 tokens for next token generation)
- **Objective Function**: Cross Entropy with mean reduction
- **vocabulary size**: 250,680

- **tokenization algorithm**: byte-level Byte Pair Encoding (BPE)

4. Training Process

By freezing most of the pretrained large language model's (LLM's) parameters and just fine-tuning a select few additional model parameters, Parameter efficient fine tunining (PEFT) techniques significantly reduce computing and storage expenses [5]. This also resolves the problem of catastrophic forgetting, which is a behavior that arises when large language models (LLMs) are fully fine-tuned [5]. Additionally, PEFT techniques have been demonstrated to generalize better to out-of-domain circumstances and outperform fine-tuning in low-data regimes [5]. In this solution, the parameter efficient fine tuning (PEFT) method used is low rank adaptation (LoRA). LoRA's method involves using low-rank decomposition to express the weight updates with two smaller matrices, or update matrices, in order to increase the efficiency of fine-tuning [6]. It is possible to train these new matrices to adjust to the new data while minimizing the total number of modifications [6]. There are no more changes made to the initial weight matrix; it stays fixed. Both the original and modified weights are summed to get the final values [6]. Below are the LoRA configuration details:

- Rank (r): 8
- LoRA alpha: 16
- target modules to apply the LoRA update matrices: **query_key_value**
- LoRA dropout: 0.05

In the configuration above,

- Rank (r): The rank of the update matrices, expressed in int. Lower rank results in smaller update matrices with fewer trainable parameters.
- LoRA alpha: LoRA scaling factor.
- LoRA dropout: dropout probability of the LoRA layers
- **query_key_value** from self attention was set as the target module to train which has the following attributes:
    - Linear layers(8 bit because of quantization)
    - input features: 2560
    - output featues: 7680
    - bias: True

Finally, after the above configurations, the parameters for training the model are as follows:

- trainable params: 2,457,600
- all params: 3,005,015,040
- trainable %: 0.0818

Additionally the important training arguments used for the trainer are as follows:

- warm up steps: 100
- max steps: 800 (0.13 epochs)
- learning rate: 1e-3
- fp16 precision: True

5. Evaluation

For evaluation of the model, again using the same methods as those used for baseline and sequence-to-sequence language model (seq2seqLM), t5-small are as follows:

- Average ROUGE-1: 0.592
- Average ROUGE-2: 0.357
- Average ROUGE-L: 0.572
- BERT embedding similarity: 0.94
- Toxicity classifier true toxic labels count: 503

6. Results

The above evaluation results show that the model performs better than the rest of the solutions. Most importantly, the true toxic labels in the model-generated detoxified texts have found to be **503** while t5-small had **602**, and the human annotated dataset had **355**. Hence, 148 more labels are still toxic. However, taking into account that the model F1-score for the toxicity model was 0.95, and since the same model is used to evaluate all, it is safe to assume that the results are uniform. Moreover, it signifies that **human** error rate is around **35 % t5-small** error rate is around **60 %** while the **bloom-3b 8bit quantized low rank adapater** model has the error rate of around **50 %** based on 1000 test samples.

7. Conclusion

Based on the observations and results from all the methods, the best evaluation method found to compare all the methods was **Toxicity true label counts using the toxicity classifier model** because using metrics like ROUGEs, which compare unigrams, can mislead, as it was seen in the baseline case where removing just the curse word makes the sentence sometimes meaningless, yet because of the absence of a single unigram, the ROUGE score isn't heavily decreased. Moreover, for BERT semantic similarity, it was seen that all the models were above 0.90, and it is misleading as well since there can be many synonyms for a single toxic word, which does not necessarily detoxify the sentence but just paraphrases it or uses a synonym instead, which might be equally toxic. Therefore, only the toxicity classification model's true label count on the model-generated detoxified texts can be considered for model performance analysis, except for the baseline dictionary approach, because it might have less toxic sentences by removing the toxic words, but it is highly prone to make sentences incomplete, and/or meaningless. The figure 1 below is the plot

of different metrics of the bloom 3b 8bit lora adapter model.

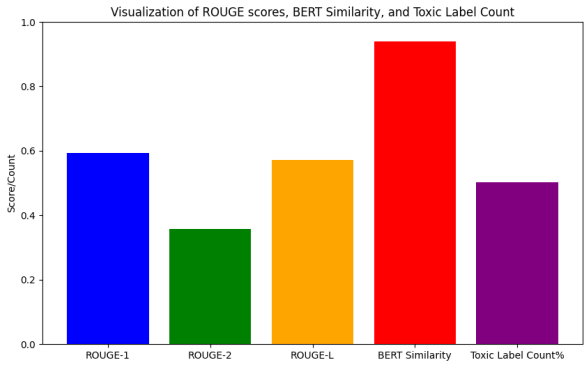

Fig 1.

The toxicity classifier **PolyGuard** model with **0.95 F1 score** can be found on huggingface under **Jayveersinh-Raj/PolyGuard**. Following the link below:
https://huggingface.co/Jayveersinh-Raj/PolyGuard

8. Future directions

It can be seen that prompt tuning a language model yields better results. Because of the huge knowledge base of the language, large language models can be prompt tuned to achieve tasks with high performance. However, due to computational limitations the model was 8-bit quantized, a model with higher parameters, and more precision would yield better results. According to [5] the potential of the method used is vast, the pretrained large language model can be topped with the tiny trained weights from parameter efficient fine tuning (PEFT) techniques. Thus, by adding tiny weights, the same large language model can be used for numerous tasks without requiring a complete model replacement.

9. References links

1. https://huggingface.co/docs/peft/task_guides/clm-prompt-tuning
2. https://towardsdatascience.com/leveraging-qlora-for-fine-tuning-of-task-fine-tuned-models-without-catastrophic-forgetting-d9bcd594cff4 ˜:text=Approaches%20Low%2DRank%20Adaptation%20(LoRA,trainable%20parameters%20for%20downstream%20tasks.
3. https://huggingface.co/docs/datasets/v1.16.1/index.html
4. https://huggingface.co/docs/optimum/concept_guides/quantization
5. https://huggingface.co/blog/peft
6. https://huggingface.co/docs/peft/conceptual_guides/lora