Deep Neural Model for Detecting Gender Stereotypes in Text

Project-II report submitted to
Indian Institute of Technology,
in partial fulfilment of the requirements
for the degree of

Bachelor of Technology (Hons.)

in

Mechanical Engineering

by

Adarsh Sharma

(20ME3AI10)

Under the supervision of

Prof. Jiaul H. Paik

Center of Excellence in Artificial Intelligence

Indian Institute of Technology Kharagpur Spring Semester, 2023-24

Certificate

This is to certify that the project report entitled "**Deep neural model for detecting gender stereotypes in text**" submitted by **Adarsh Sharma** (Roll No.: 20ME3AI10) to the Indian Institute of Technology, Kharagpur towards the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology (Hons.) inMechanical Engineering of the Institute, is a record of bona fide work carried out by him undermy supervision and guidance during the Spring Semester of 2024.

Professor: Jiaul H. Paik

Date: 30th April 2024 CenterOfExcellenceInArtificialIntelligence

Indian

Institute of Technology, Kharagpur

Kharagpur - 721302, India

Declaration

I certify that:

a) The work contained in this report is original and has been done by me under the guidance of my supervisor.

b) The work has not been submitted to any other Institute for any degree or diploma.

c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

d) I have adhered to the guidelines provided by the Institute in preparing the report.

e) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from thecopyright owners of the sources, whenever necessary.

Date: 30/04/2024 Adarsh Sharma

Place: Kharagpur (20ME3AI10)

Acknowledgement

I extend my sincere appreciation to my supervisor, **Prof. Jiaul H. Paik**, and my guide, for their invaluable support and guidance during my B. Tech thesis project. Their unwavering encouragement, insightful feedback, and constructive criticism have played a pivotal role in shaping my research and assisting me in achieving my academic aspirations. The wealth of knowledge and expertise they bring to the field has served as a constant source of inspiration, fostering a deeper understanding of the subject matter.

I am grateful to the **Indian Institute of Technology Kharagpur** for providing the essential resources and facilities that facilitated the execution of my research work. My heartfelt thanks also go to my family and friends for their consistent support and encouragement throughout my academic journey.

Finally, I express my sincere gratitude to all those who have contributed to my research in various capacities. Your collective support and guidance have been instrumental in the success of this endeavor.

Thank you all for being a crucial part of my academic journey.

Adarsh Sharma

Table of Content

\

5.Abstract

In this project, I addressed the task of text classification aimed at identifying gender stereotypes within textual data. Leveraging the state-of-the-art language model (LLM) Gemma, I employed the Quantized Low rank adaptation (QLoRA) technique for fine-tuning. QLoRA, known for its efficiency in reducing VRAM consumption while maintaining competitive accuracy, enabled us to achieve a classification accuracy of 95.2%. Our methodology involved training Gemma on a dataset consisting of both stereotype and non-stereotype text samples, allowing the model to discern between the two categories effectively. Looking forward, I aim to explore other parameter-efficient fine-tuning techniques such as prefix tuning to enhance the efficiency and performance of our model further. Additionally, I intend to delve into methodologies like Reinforcement Learning with Human Feedback (RLHF) under the guidance of Prof. PK Mishka for our future work. RLHF presents an exciting avenue for incorporating human feedback into the model training process, potentially leading to even more nuanced and accurate classifications of gender stereotypes in textual data. This project on text classification aimed at identifying gender stereotypes within textual data not only contributes to the advancement of text classification techniques but also holds implications for addressing gender biases and stereotypes present in language data, thereby fostering more inclusive and equitable language processing systems.

Chapter 1

1.1 Introduction

Language, the cornerstone of human interaction, serves not only as a means of communication but also as a mirror reflecting societal biases. Embedded within our everyday speech and writing can lie stereotypes – ingrained assumptions and generalizations about social groups. These stereotypes, often unconscious, can have a profound impact on how we perceive and interact with the world around us. They can perpetuate negative portrayals, limit opportunities, and hinder effective communication, ultimately reinforcing social inequalities.

This project tackles the challenge of identifying stereotypes in text data through the power of deep neural models. We aim to develop a sophisticated model capable of detecting stereotypical language related to gender, race, ethnicity, or other social categories. By analyzing a broad spectrum of textual formats, ranging from news articles and social media posts to marketing materials and fictional narratives, the model will be trained to recognize patterns indicative of stereotypical language. This includes identifying the use of specific words, phrases, and sentence structures that reinforce biased assumptions.

The ability to automatically flag potentially biased language offers a powerful tool for fostering positive change. It can raise awareness of unconscious stereotypes that may permeate our communication, prompting individuals and organizations to critically examine their language choices. Consider a company advertisement that unintentionally reinforces gender stereotypes by portraying women primarily in domestic roles. A well-trained model could identify such bias, prompting the company to revise the advertisement to promote a more inclusive and equitable image. Similarly, a social media platform could leverage such technology to flag comments containing racial stereotypes, fostering a more respectful online environment.

Furthermore, this project holds significant implications for promoting diversity and inclusion efforts. By identifying and flagging stereotypical language, we can encourage a more mindful approach to communication, fostering a society that embraces differences and celebrates the richness of human experience. The model can serve as a valuable tool for educators and organizations working towards creating a fairer and more inclusive world. By integrating this technology into educational settings, for

example, students can develop a heightened awareness of stereotypical language, fostering critical thinking skills and promoting a more empathetic and inclusive discourse.

Ultimately, this project aspires to contribute to a more equitable and respectful communication landscape. By harnessing the power of deep neural models to identify stereotypes in text data, we can raise awareness of unconscious biases and encourage the adoption of inclusive language practices. This, in turn, can pave the way for a more just and harmonious society, where communication fosters understanding and celebrates the diversity of human experience.

Chapter 2

Datasets:

For our sentiment analysis project, I trained and tested the model on two datasets. The first, a publicly available collection of IMDb movie reviews, provided a broad base of labeled sentiment. The second dataset, custom-built for this project, included both stereotypical and non-stereotypical sentences.

1. IMDb review dataset:

The IMDb dataset is a popular resource for machine learning tasks, especially sentiment analysis. It contains a collection of movie reviews labeled as positive or negative. Some versions offer 50,000 reviews for training and testing. This dataset helps train algorithms to classify text data. Here we have taken a subset of IMDb dataset for training and testing purposes. The training dataset contains 1000 rows with 51% positive reviews and 49% negative reviews, whereas testing dataset contains 600 rows with 51% positve reviews and 49% negative reviews.

```
Training Dataset:
        Positive Reviews: 501
        Negative Reviews: 500


Test Dataset:
        Positive Reviews: 296
        Negative Reviews: 304
```

2. Custom Stereotype dataset:

2.1 Data Composition:

The dataset consists of a collection of sentences categorized as either stereotypical or non-stereotypical regarding gender roles. Here are some examples:

Stereotypical Sentences:
"A housewife cleans the house."

"Men are the breadwinners of the family."
"Women are not good at math."

Non-Stereotypical Sentences:
"Everyone helps with chores in our household."
"Both parents can have successful careers."
"Anyone can be good at math regardless of gender."

The dataset is designed to provide a balanced representation of both stereotypical and non-stereotypical sentences to train a robust classification model.
The dataset has been divided into 5 parts (df1 to df5), each containing 74 rows. Each row consists of 24 sentences that display a stereotype and 50 sentences that do not. To train the model on this dataset, I have applied 5-fold cross-validation. In each fold, I trained the model on 4 parts of the dataset and tested it on the remaining part.

```
Training Dataset:
        Positive Sentences: 96
        Negative Sentences: 200

Test Dataset:
        Positive Sentences: 24
        Negative Sentences: 50
```

2. 2 Data cleaning:

The process of data cleaning involves removing punctuation marks and converting all words to lowercase letters.

Punctuation symbols considered are:

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

- **Generally helpful:** Removing punctuation can improve model performance by treating for e.g. "data" and "data!" the same.
- **Exceptions exist:** However, For sentiment analysis, punctuation like exclamation points might be important.

I trained the model with punctuation and without punctuation and found that slightly more accuracy was seen when with punctuation. So, in the final model with higher epochs I decided to have punctuations

Lowercasing a dataset for NLP model training has both pros and cons:

Pros:

- **Reduces data sparsity:** "He" and "he" are treated as the same word, reducing the number of

unique words the model needs to learn. This is especially helpful with smaller datasets.

- **Improves model consistency:** The model doesn't need to learn the meaning based on capitalization, making it more consistent.

Cons:

- **Loses information:** In some tasks, capitalization can be important. For example, in Named Entity Recognition (NER), "Paris" (city) and "paris" (brand) are different entities.

I trained the model on both cases and found and found that without lowercasing provided slight better results for our dataset. So, I stuck with no lowercasing for the final model with higher (=10) epochs.

Other techniques like stopwords removal were applied but that didn't give good results, most probably because short sentences already have fewer words. Removing stopwords might leave too little information for the model to learn from.

1. Before stopwords removal:

```
Out[5]:
'Melani with her daughter.'
```

2. After stopwords removal:

```
Out[7]:
'melani daughter'
```

Chapter 3

1. History of LLMs

Language Model (LM) advancements have long been a frontier in artificial intelligence research. The journey toward sophisticated Language Model models (LLMs) has been marked by significant milestones and innovations, each contributing to the evolution of these powerful tools.

The inception of LLMs can be traced back to the early days of natural language processing (NLP) research. Initial attempts to understand and process human language relied on rule-based systems, which struggled to capture the nuances and complexities of language. However, the advent of statistical methods and machine learning algorithms revolutionized the field, paving the way for more data-driven approaches.

One of the key breakthroughs in the development of LLMs was the introduction of neural network architectures, particularly recurrent neural networks (RNNs) and long short-term memory networks (LSTMs). These architectures enabled models to learn sequential patterns and dependencies in language data, allowing for more nuanced and context-aware language processing.

The emergence of large-scale datasets, such as the Common Crawl and Wikipedia dumps, provided the necessary fuel for training increasingly complex LLMs. With access to vast amounts of text data, researchers could train models on a diverse range of linguistic patterns and structures, improving their ability to understand and generate human-like text.

However, the true turning point in the evolution of LLMs came with the advent of transformer architectures, most notably exemplified by models like OpenAI's GPT (Generative Pre-trained Transformer) series. Transformers introduced a novel self-attention mechanism that enabled models to capture long-range dependencies in text more effectively, surpassing the limitations of earlier architectures.

Pre-training became a central paradigm in LLM development, where models were first trained on large corpora of text data in an unsupervised manner, followed by fine-tuning on specific downstream tasks. This approach allowed LLMs to leverage the vast amount of unlabeled text data available on the internet, resulting in significant performance gains across a wide range of NLP tasks.

The release of large-scale pre-trained models like GPT-2 and GPT-3 marked a milestone in the democratization of LLM technology, making state-of-the-art language understanding and generation

capabilities accessible to a broader audience. These models demonstrated unprecedented proficiency in tasks such as language translation, text summarization, question answering, and even creative writing.

Moreover, the versatility of LLMs has led to their adoption across various domains and industries, including healthcare, finance, customer service, and entertainment. From aiding medical diagnosis to automating customer support interactions, LLMs have shown immense potential to transform how we interact with and leverage textual data.

Looking ahead, the journey of LLMs is poised to continue, with ongoing research efforts focused on improving model efficiency, scalability, and interpretability. As these technologies become more refined and accessible, they hold the promise of further revolutionizing how we communicate, work, and interact with information in the digital age.

2. Why fine tuning LLMs:

Fine-tuning is taking a pre-trained model and training at least one internal model parameter (i.e. weights). In the context of LLMs, what this typically accomplishes is transforming a general-purpose base model (e.g. GPT-3) into a specialized model for a particular use case (e.g. ChatGPT)

The key upside of this approach is that models can achieve better performance while requiring (far) fewer manually labeled examples compared to models that solely rely on supervised training.

While strictly self-supervised base models can exhibit impressive performance on a wide variety of tasks with the help of prompt engineering, they are still word predictors and may generate completions that are not entirely helpful or accurate.

Fine-tuning data for Language Models (LLMs) like Gemma is crucial for several reasons. While LLMs are remarkably powerful and can generate coherent text based on input prompts, direct inferencing without fine-tuning may not always yield optimal results.

Firstly, fine-tuning allows customization of the model to suit specific tasks or domains. Pre-trained LLMs like Gemma are trained on a diverse range of text from the internet, covering various topics and writing styles. However, this generic training may not capture the nuances of a particular domain. Fine-tuning with domain-specific data enables the model to better understand the context, vocabulary, and conventions relevant to that domain. For example, fine-tuning a model on medical literature can enhance its ability to generate accurate medical text.

3. Ways to fine tune LLMs:

There are 3 generic ways one can fine-tune a model: self-supervised, supervised, and reinforcement learning. These are not mutually exclusive in that any combination of these three approaches can be used in succession to fine-tune a single model.

1. Self-supervised learning:

Self-supervised learning consists of training a model based on the inherent structure of the training data. In the context of LLMs, what this typically looks like is given a sequence of words (or tokens, to be more precise), predict the next word (token).

While this is how many pre-trained language models are developed these days, it can also be used for model fine-tuning. A potential use case of this is developing a model that can mimic a person's writing style given a set of example texts.

2. Reinforcement learning:

We can use reinforcement learning (RL) to fine-tune models. RL uses a reward model to guide the training of the base model. This can take many different forms, but the basic idea is to train the reward model to score language model completions such that they reflect the preferences of human labelers. The reward model can then be combined with a reinforcement learning algorithm (e.g. Proximal Policy Optimization (PPO)) to fine-tune the pre-trained model.
An example of how RL can be used for model fine-tuning is demonstrated by OpenAI's InstructGPT models, which were developed through 3 key steps:

1. Generate high-quality prompt-response pairs and fine-tune a pre-trained model using supervised learning. (~13k training prompts) Note: One can (alternatively) skip to step 2 with the pre-trained model [3].
2. Use the fine-tuned model to generate completions and have human-labelers rank responses based on their preferences. Use these preferences to train the reward model. (~33k training prompts)
3. Use the reward model and an RL algorithm (e.g. PPO) to fine-tune the model further. (~31k training prompts)

While the strategy above does generally result in LLM completions that are significantly more preferable to the base model, it can also come at a cost of lower performance in a subset of tasks. This drop in performance is also known as an alignment tax.

3. Supervised learning:

As we saw above, there are many ways in which one can fine-tune an existing language model. However, for the remainder of this article, we will focus on fine-tuning via supervised learning.

Options for Parameter Training in supervised learning:

1. Retrain all paramters:

The first option is to train all internal model parameters (called full parameter tuning) [3]. While this option is simple (conceptually), it is the most computationally expensive. Also, the phenomenon of catastrophic forgetting where the model forgets what it learned in the initial stages is also observed in this case.

```
OutOfMemoryError: CUDA out of memory. Tried to allocate 1.08 GiB. GPU 0 has a total capact
y of 15.89 GiB of which 778.12 MiB is free. Process 3470 has 15.00 GiB memory in use. Of t
he allocated memory 12.86 GiB is allocated by PyTorch, and 1.85 GiB is reserved by PyTorch
but unallocated. If reserved but unallocated memory is large try setting max_split_size_mb
to avoid fragmentation.  See documentation for Memory Management and PYTORCH_CUDA_ALLOC_CO
NF
```

2. Traning last layers of the Pretrained LLMs:

The idea with this is to preserve the useful representations/features the model has learned from past training when applying the model to a new task. This generally consists of dropping "the head" of a neural network (NN) and replacing it with a new one (e.g. adding new layers with randomized weights). While leaving the majority of parameters untouched mitigates the huge computational cost of training an LLM, TL may not necessarily resolve the problem of catastrophic forgetting as per most of the sources I referred to.

3. Parameter efficient fine tuning:

Techniques like QLoRA, LoRA, prefix tuning etc are used which reduces the no. of parameter for training.

1. Prefix tuning

Prefix tuning is a technique for adapting large language models (LLMs) to specific tasks. It works by adding a short piece of text, called a prefix, to the beginning of the input. This prefix is like a hint that guides the LLM towards the desired outcome. Unlike traditional fine-tuning, which modifies the entire LLM, prefix tuning only adjusts the prefix itself, making it more efficient and requiring less storage space. **(**In future i will try fine tuning using prefix tuning)

2. **QLoRA:**

The QLoRA paper proposes a method for efficiently fine-tuning large language models (LLMs) on a single GPU. Fine-tuning is a crucial step in adapting a pre-trained LLM to a specific task, but it can be computationally expensive due to the massive size of these models. QLoRA addresses this challenge by combining three key techniques:

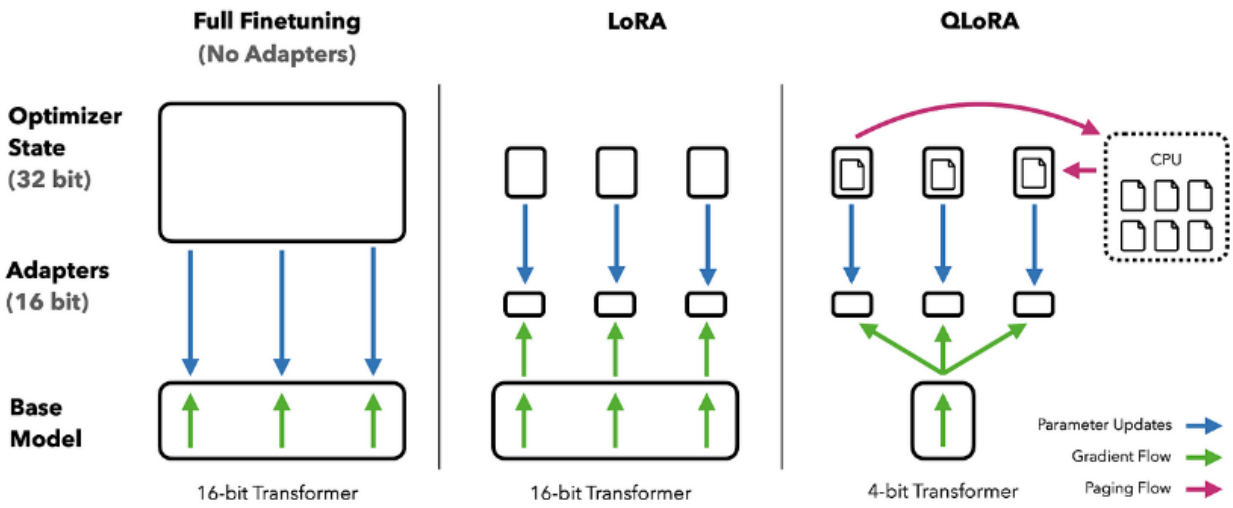
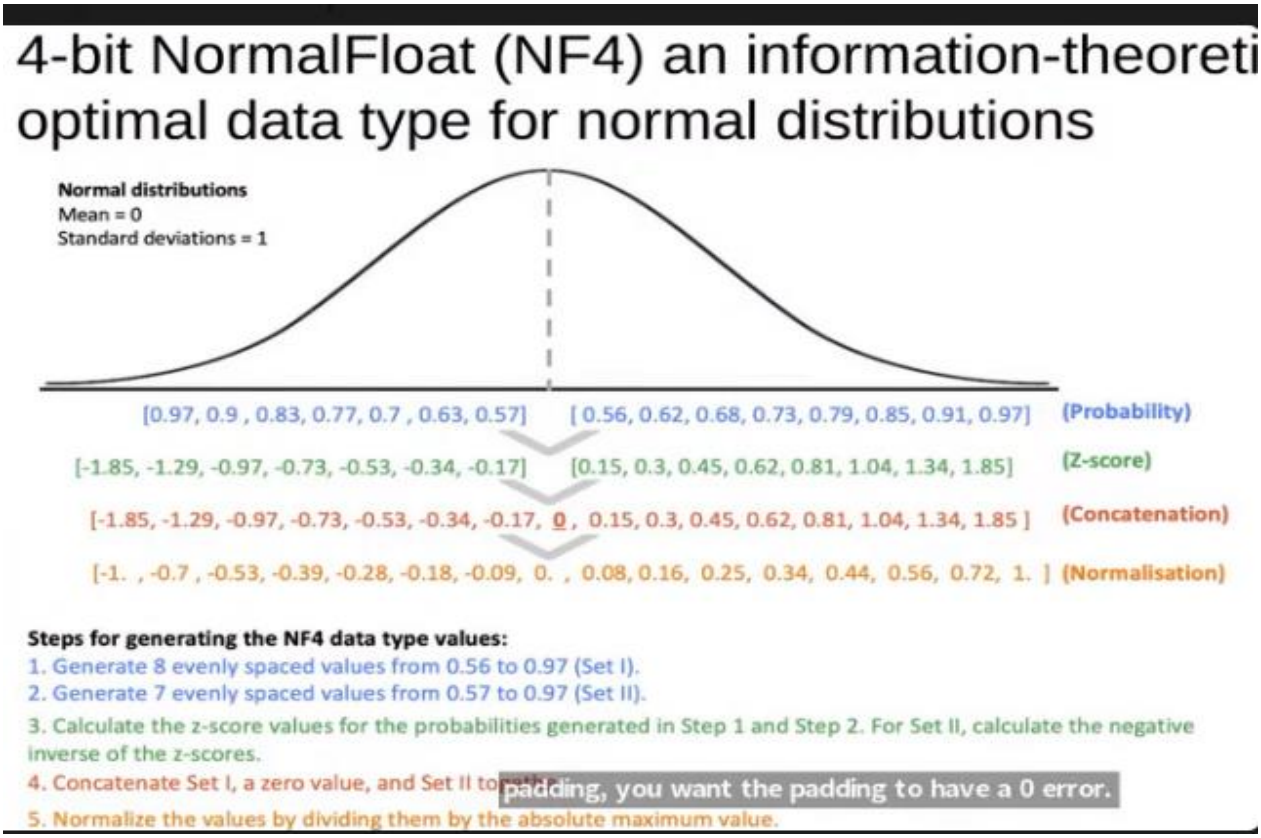


Figure 1: Different finetuning methods and their memory requirements. QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.

Quantization: This technique reduces the precision of the model's weights from 32 bits (single-precision floating-point) to a lower precision format, typically 4 bits. This significantly reduces memory consumption, allowing the entire model to fit on a single GPU.

4-bit NormalFloat (NF4) Quantization: A specific type of quantization, NF4, is used in QLoRA. It offers better accuracy compared to standard 4-bit quantization by handling outliers in the data more effectively.



Double Quantization: This technique further optimizes memory usage by quantizing the quantization constants themselves. These constants are used to scale the weights during the quantization process. By quantizing them, QLoRA saves additional memory without sacrificing accuracy.

Paged Optimizers: Fine-tuning large models often leads to memory spikes during backpropagation, a key step in training. QLoRA tackles this issue by introducing Paged Optimizers. This technique leverages NVIDIA's unified memory to manage memory allocation efficiently, preventing out-of-memory errors and enabling training on a single GPU.

LoRA (Low-Rank Adaptation of Large Language Models) is a popular and lightweight training technique that significantly reduces the number of trainable parameters. It works by inserting a smaller number of new weights into the model and only these are trained. This makes training with LoRA much faster, memory-efficient, and produces smaller model weights (a few hundred MBs), which are easier to store and share.

$W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d \times r}, A \in \mathbb{R}^{r \times k}$,

$W_0 \in \mathbb{R}^{d \times k}$,

Benefits of QLoRA:

1. **Reduced Memory Footprint:** By quantizing weights and using Paged Optimizers, QLoRA allows fine-tuning large models on a single GPU which would otherwise require multiple GPUs or specialized hardware.

2. **Cost-Effective Training:** Reduced hardware requirements lead to lower training costs.

Overall, QLoRA presents a significant advancement in fine-tuning large language models. By combining quantization techniques and Paged Optimizers, it enables efficient training on a single GPU, opening doors for wider adoption and application of LLMs.

Chapter 4

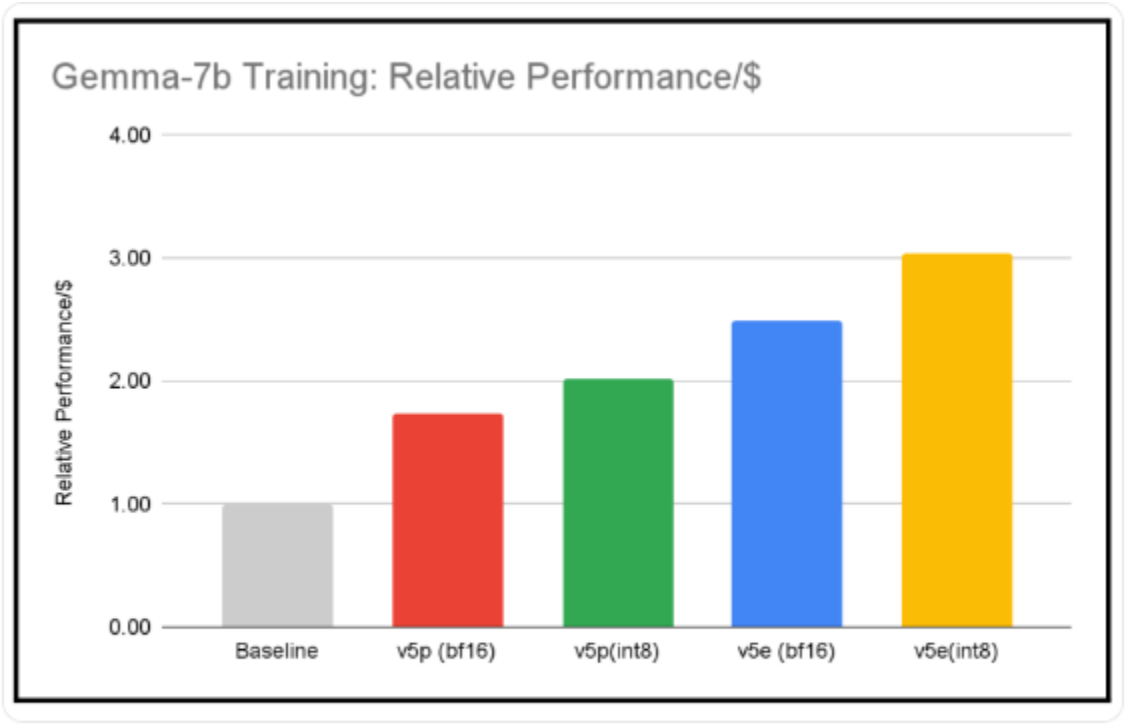
Gemma:

Developed by Google DeepMind, Gemma is a family of cutting-edge LLMs designed to democratize access to powerful AI technology. Inspired by the "Gemini" research project, its name reflects a "precious stone," signifying its potential value.

Gemma offers a unique advantage: it's open-source. This means researchers and developers worldwide can explore its capabilities, contribute to its development, and tailor it for specific applications.

Gemma comes in two sizes: 2B and 7B parameters, catering to different needs. The smaller size is ideal for resource-constrained environments, while the larger one tackles complex tasks. Each size also has "base" and "instruction-tuned" variants, offering flexibility for various purposes.

While the Gemma 7B model leverages a multihead attention mechanism, Gemma 2B utilizes multi-query attention. This approach aids in reducing memory bandwidth requirements during the inference process, which can potentially be advantageous for Gemma 2B on-device inference scenarios, where memory bandwidth is often limited.



Gemma-7b and baseline relative training performance per dollar. Measured using Gemma 7B (MaxText) on TPU v5e-256 and v5p-128. context length 8192. Baseline (LLama2-7b) performance is derived using total GPU hours and total number of training tokens as per the published results. Performance/$ is derived using the list price of respective accelerators. As of Feb, 2024.

We derived performance per dollar using (peak-flops*EMFU)/ (list price of VM instance). Using the MaxText reference implementation, we observed up to 3X better performance per dollar for the Gemma 7B model with respect to the baseline training performance (Llama2 7B). Please note that the performance or performance-per-dollar differences presented here are functions of the model architecture, hyperparameters, the underlying accelerator and training software; better performance results cannot be solely attributed to any of these factors alone.

I experimented with different variants of gemma and found that “gemma/transformers/7b-it/1” was one of the best among the other gemma models that were shared by google.

Chapter 5

Configurations:

1. BitsAndBytesConfig:

- **load_in_4bit**=True,

We have 2 options load in 4bit or 8bit(with int8 datatype) i have done it in 4bit as it will reduce memory usage and it peforms better than int8

- **bnb_4bit_use_double_quant=** false

Through in QLoRA paper it was observed that using double quantization was not affecting model performance significantly but i still didn't used it as it is basically a memory saving technique and my model didn’t showed memory limit exceeded even without using it. But if model shows out of memory error having double quantization can be helpful.

- **bnb_4bit_quant_type="**nf4**"**

We can use either 'nf4' which is new datatype proposed in the QLoRAer or we can use standard fp4 representation, we have used "nf4" because in the QLoRA research paper they observed that nf4 performed better than fp4, I will be definitely in future, compare the result of fp4, nf4 and int8 here.

```
compute_dtype = getattr(torch, "float16")

bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_use_double_quant=False,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=compute_dtype,
)
```

2. Prompt given to model:

1. Prompt given while training:

```
generate_prompt
          Analyze the stereotype of the sentence enclosed in square brackets,
          determine if it is positive, or negative, and return the answer as
          the corresponding stereotype label "positive" or "negative"

          [Rivers , like the Colorado River , carry enormous loads of sand and soil that is picked up from erosional processes
.] = Negative<eos>
```

2. Prompt given while testing

```
Analyze the stereotype of the sentence enclosed in square brackets,
           determine if it is positive,or negative, and return the answer as
           the corresponding stereotype label "positive" or "negative"

           [Turbine Fig . Each turbine is made of curved blades arranged like the sails of a windmill .] =
```

3. LoRA configuration and arguments to SFTTrainer:

- lora_alpha=16

[...] and LoRA alpha is the scaling factor for the weight matrices. The weight matrix is scaled by $\frac{lora\_alpha}{lora\_rank}$, and a higher alpha value assigns more weight to the LoRA activations. We chose 16 since this was common practice in training scripts we reviewed and chose a 1:1 ratio so as not to overpower the base model.

- Rank of update matrices =64

I checked model performance for the rank of 32 as well but, model performed well on 64 rank.

```
peft_config = LoraConfig(
    lora_alpha=16,
    lora_dropout=0,
    r=64,
    bias="none",
    task_type="CAUSAL_LM",
    target_modules=["q_proj", "k_proj", "v_proj", "o_proj",
                    "gate_proj", "up_proj", "down_proj",],
)

training_arguments = TrainingArguments(
    output_dir="logs",
    num_train_epochs=10,
    gradient_checkpointing=True,
    per_device_train_batch_size=1,
    gradient_accumulation_steps=8,
    optim="paged_adamw_32bit",
    save_steps=0,
    logging_steps=25,
    learning_rate=2e-4,
    weight_decay=0.001,
    fp16=True,
    bf16=False,
    max_grad_norm=0.3,
    max_steps=-1,
    warmup_ratio=0.03,
    group_by_length=False,
    evaluation_strategy="no",                # no evaluation done
    lr_scheduler_type="cosine",
    report_to="tensorboard",
)

trainer = SFTTrainer(
    model=model,
    train_dataset=train_data,
    peft_config=peft_config,
    dataset_text_field="text",
    tokenizer=tokenizer,
    max_seq_length=max_seq_length,
    args=training_arguments,
    packing=False,
)
```

Chapter-6

Results:

1. On IMDb dataset:

1. 1 Without fine tuning direct inference:

```
Accuracy: 0.427
Accuracy for label 0: 0.157
Accuracy for label 1: 1.000

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.16      0.27        51
           1       0.36      1.00      0.53        24

    accuracy                           0.43        75
   macro avg       0.68      0.58      0.40        75
weighted avg       0.79      0.43      0.35        75


Confusion Matrix:
[[ 8 43]
 [ 0 24]]
```

1.2 With fine tuning:

```
Accuracy: 0.920
Accuracy for label 0: 0.980
Accuracy for label 1: 0.792

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.98      0.94        51
           1       0.95      0.79      0.86        24

    accuracy                           0.92        75
   macro avg       0.93      0.89      0.90        75
weighted avg       0.92      0.92      0.92        75


Confusion Matrix:
[[50  1]
 [ 5 19]]
```

2. On gender stereotype dataset:

We have divided a dataset of 370 elements into 5 equal parts (df1, df2, df3, df4, df5) with an equal ratio of positive and negative cases in each part. Here label 1 means positive and 0 means negative

2.1. Without fine tuning direct inference:

A. Train dataset df1,df2,df3,df4 an test dataset is df0:

```
Accuracy: 0.520
Accuracy for label 0: 0.569
Accuracy for label 1: 0.417

Classification Report:
              precision    recall  f1-score   support

           0       0.67      0.57      0.62        51
           1       0.31      0.42      0.36        24

    accuracy                           0.52        75
   macro avg       0.49      0.49      0.49        75
weighted avg       0.56      0.52      0.53        75


Confusion Matrix:
[[29 22]
 [14 10]]
```

B. Train dataset df0,df2,df3,df4 an test dataset is df1:

```
Accuracy: 0.547
Accuracy for label 0: 0.588
Accuracy for label 1: 0.458

Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.59      0.64        51
           1       0.34      0.46      0.39        24

    accuracy                           0.55        75
   macro avg       0.52      0.52      0.52        75
weighted avg       0.58      0.55      0.56        75


Confusion Matrix:
[[30 21]
 [13 11]]
```

C. Train dataset df0,df1,df3,df4 an test dataset is df2:

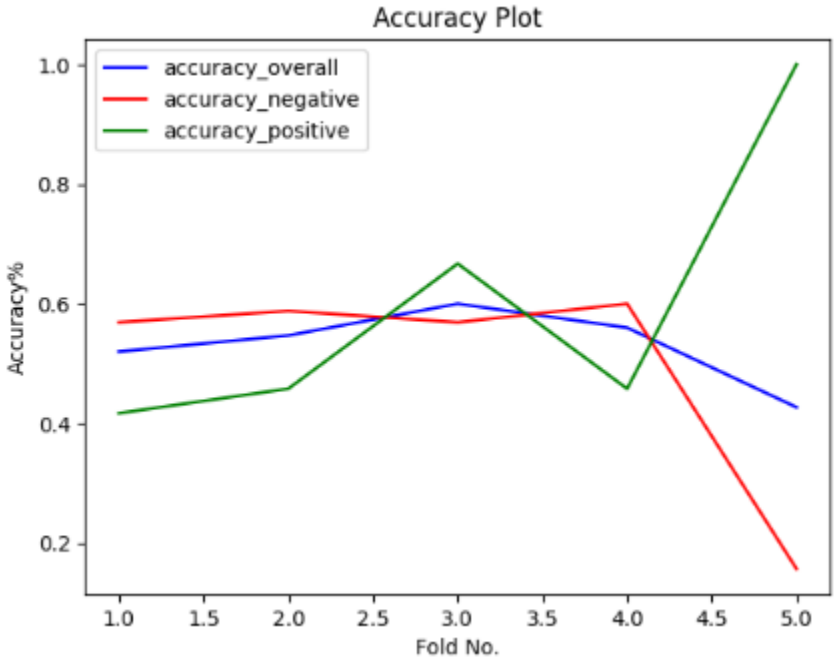
```
Accuracy: 0.600
Accuracy for label 0: 0.569
Accuracy for label 1: 0.667

Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.57      0.66        51
           1       0.42      0.67      0.52        24

    accuracy                           0.60        75
   macro avg       0.60      0.62      0.59        75
weighted avg       0.67      0.60      0.61        75


Confusion Matrix:
[[29 22]
 [ 8 16]]
```

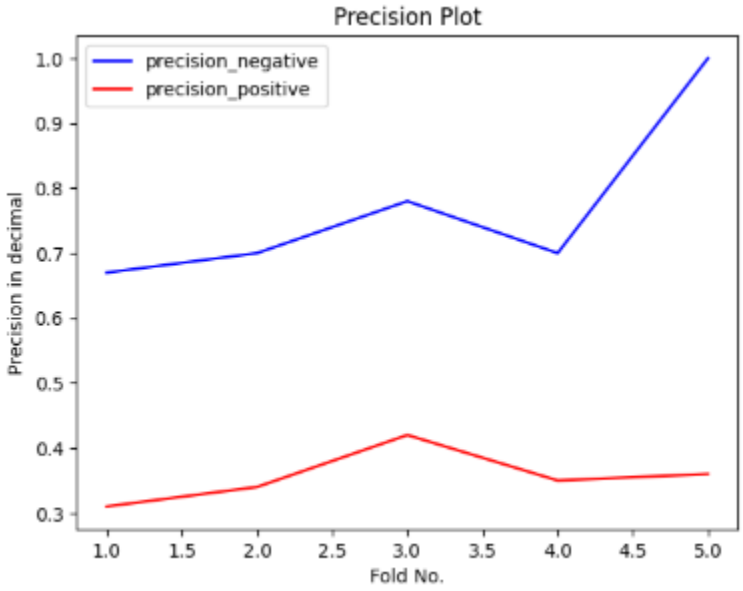
D. Train dataset df0,df1,df2,df4 an test dataset is df3:

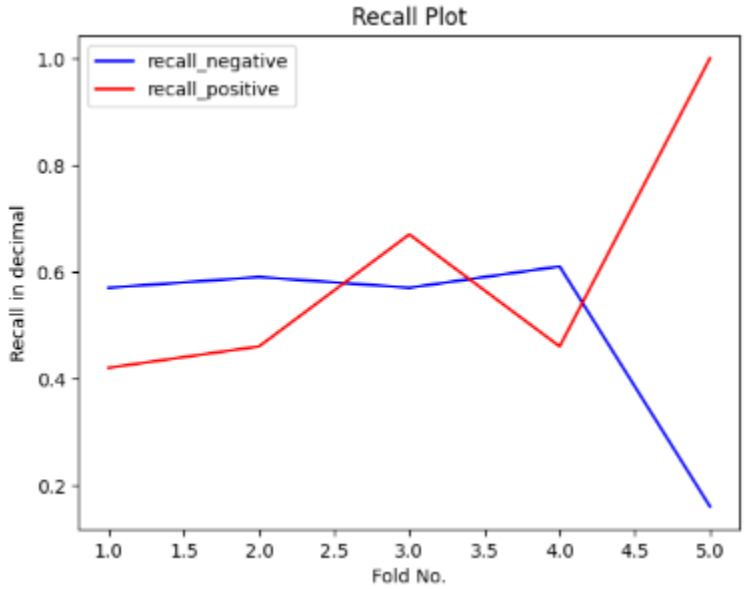
```
Accuracy: 0.560
Accuracy for label 0: 0.608
Accuracy for label 1: 0.458

Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.61      0.65        51
           1       0.35      0.46      0.40        24

    accuracy                           0.56        75
   macro avg       0.53      0.53      0.53        75
weighted avg       0.59      0.56      0.57        75


Confusion Matrix:
[[31 20]
 [13 11]]
```

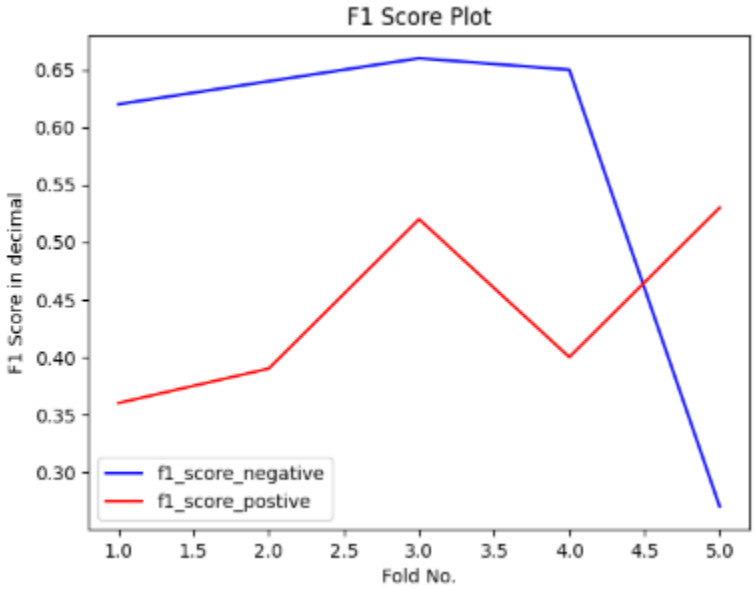
E. Train dataset df0,df1,df2,df3 an test dataset is df4:

```
Accuracy: 0.427
Accuracy for label 0: 0.157
Accuracy for label 1: 1.000

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.16      0.27        51
           1       0.36      1.00      0.53        24

    accuracy                           0.43        75
   macro avg       0.68      0.58      0.40        75
weighted avg       0.79      0.43      0.35        75


Confusion Matrix:
[[ 8 43]
 [ 0 24]]
```


Accuracy Plot
accuracy_overall
accuracy_negative
accuracy_positive
Accuracy%
Fold No.


Precision Plot
precision_negative
precision_positive
Precision in decimal
Fold No.






```
Average accuracy_overall:  0.5308
Average accuracy_negative:  0.4966000000000004
Average accuracy_positive:  0.6
Average precision_negative:  0.7700000000000001
Average precision_positive:  0.356
Average recall_negative:  0.5
Average recall_positive:  0.6020000000000001
Average f1_score_negative:  0.568
Average f1_score_postive:  0.4400000000000006
```

2.2 With fine tuning:

A. Train dataset df1,df2,df3,df4 an test dataset is df0:

Step	Training Loss
25	5.575800
50	0.713600
75	0.625000
100	0.312200
125	0.203300
150	0.158300
175	0.118700
200	0.111400
225	0.106300
250	0.088400
275	0.088300
300	0.083400
325	0.078600
350	0.078300

```
Accuracy: 0.960
Accuracy for label 0: 1.000
Accuracy for label 1: 0.875

Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        51
           1       1.00      0.88      0.93        24

    accuracy                           0.96        75
   macro avg       0.97      0.94      0.95        75
weighted avg       0.96      0.96      0.96        75


Confusion Matrix:
[[51  0]
 [ 3 21]]
```

B. Train dataset df0,df2,df3,df4 an test dataset is df1:

Step	Training Loss
25	5.639800
50	0.719000
75	0.621400
100	0.308200
125	0.200700
150	0.159800
175	0.115300
200	0.108800
225	0.102400
250	0.088700
275	0.088300
300	0.082500
325	0.078600
350	0.077700

```
Accuracy: 0.933
Accuracy for label 0: 0.941
Accuracy for label 1: 0.917

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95        51
           1       0.88      0.92      0.90        24

    accuracy                           0.93        75
   macro avg       0.92      0.93      0.92        75
weighted avg       0.93      0.93      0.93        75


Confusion Matrix:
[[48  3]
 [ 2 22]]
```

C. Train dataset df0,df1,df3,df4 an test dataset is df2:

Step	Training Loss
25	5.646700
50	0.751700
75	0.608000
100	0.312900
125	0.204600
150	0.161400
175	0.115500
200	0.108700
225	0.105200
250	0.087500
275	0.089200
300	0.083800
325	0.079100
350	0.077500

```
Accuracy: 0.947
Accuracy for label 0: 0.941
Accuracy for label 1: 0.958

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.94      0.96        51
           1       0.88      0.96      0.92        24

    accuracy                           0.95        75
   macro avg       0.93      0.95      0.94        75
weighted avg       0.95      0.95      0.95        75


Confusion Matrix:
[[48  3]
 [ 1 23]]
```

D. Train dataset df0,df1,df2,df4 an test dataset is df0:

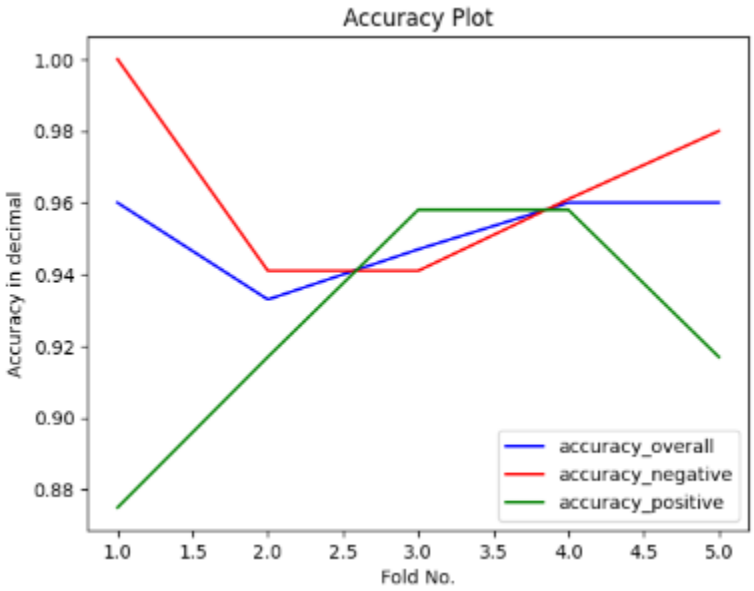
Step	Training Loss
25	5.691900
50	0.772600
75	0.612000
100	0.304000
125	0.207400
150	0.154900
175	0.118200
200	0.110100
225	0.104200
250	0.087900
275	0.086600
300	0.083800
325	0.079100
350	0.077700

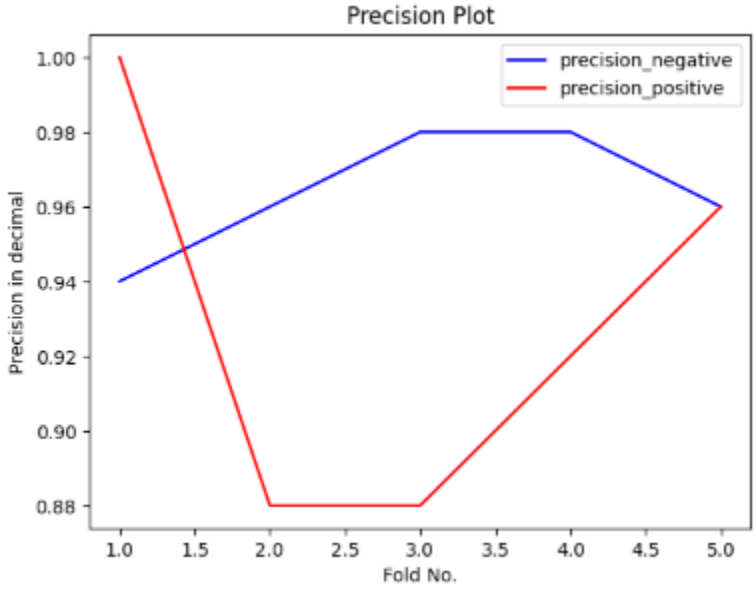
```
Accuracy: 0.960
Accuracy for label 0: 0.961
Accuracy for label 1: 0.958

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97        51
           1       0.92      0.96      0.94        24

    accuracy                           0.96        75
   macro avg       0.95      0.96      0.95        75
weighted avg       0.96      0.96      0.96        75


Confusion Matrix:
[[49  2]
 [ 1 23]]
```

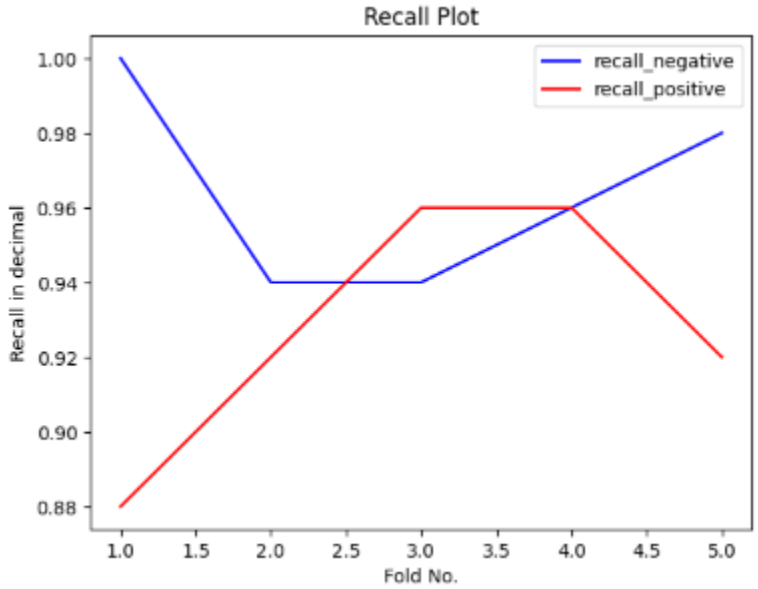
E. Train dataset df0,df1,df2,df3 an test dataset is df4:

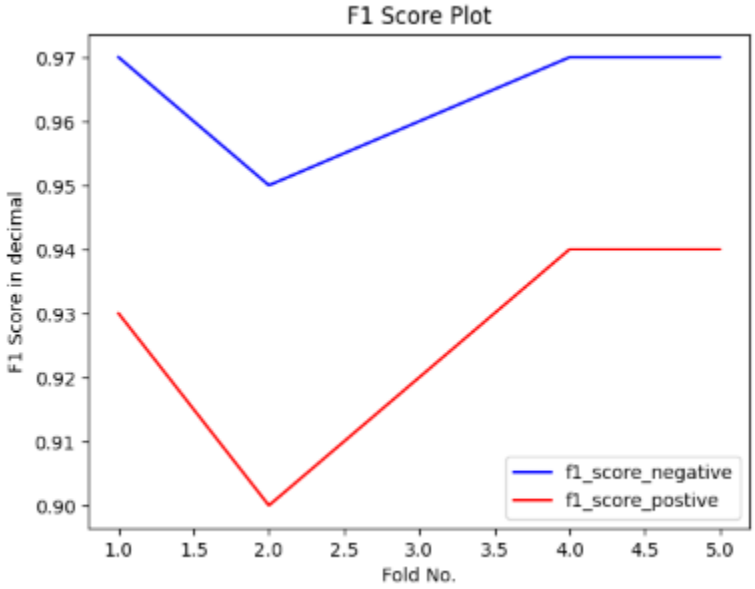
Step	Training Loss
25	6.440100
50	0.819100
75	0.632400
100	0.296200
125	0.231300
150	0.169400
175	0.129000
200	0.120700
225	0.115600
250	0.099100
275	0.096800
300	0.093300
325	0.088100
350	0.086200

```
Accuracy: 0.960
Accuracy for label 0: 0.980
Accuracy for label 1: 0.917

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97        51
           1       0.96      0.92      0.94        24

    accuracy                           0.96        75
   macro avg       0.96      0.95      0.95        75
weighted avg       0.96      0.96      0.96        75


Confusion Matrix:
[[50  1]
 [ 2 22]]
```


Accuracy Plot
Accuracy in decimal
1.00
0.98
0.96
0.94
0.92
0.90
0.88
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Fold No.
accuracy_overall
accuracy_negative
accuracy_positive


Precision Plot
Precision in decimal
1.00
0.98
0.96
0.94
0.92
0.90
0.88
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Fold No.
precision_negative
precision_positive






```
Average accuracy_overall:  0.952
Average accuracy_negative:  0.9645999999999999
Average accuracy_positive:  0.925
Average precision_negative:  0.9640000000000001
Average precision_positive:  0.9279999999999999
Average recall_negative:  0.9640000000000001
Average recall_positive:  0.9279999999999999
Average f1_score_negative:  0.9639999999999999
Average f1_score_postive:  0.9259999999999999
```

Chapter 7

Conclusion:

In this project, we embarked on the task of text classification to discern gender stereotypes within a given dataset. Leveraging the power of large language models (LLMs), specifically the Gemma model, our primary aim was to develop a robust classifier capable of accurately identifying instances of gender stereotype within textual data.

Initially, we employed prompt tuning on the Gemma LLM, a method known for its simplicity and efficiency. Despite its ease of implementation, the performance yielded was suboptimal, with an accuracy of approximately 53.78%. Recognizing the need for more nuanced fine-tuning techniques, we turned to QLoRA, a method celebrated for its effectiveness in reducing memory consumption while maintaining high accuracy levels.

Through QLoRA fine-tuning, I achieved a significant improvement in performance, attaining an average accuracy of 95.2% and precision of 92% for positive stereotype in our classification task. This substantial enhancement underscores the importance of employing sophisticated fine-tuning strategies to harness the full potential of LLMs for specialized tasks like stereotype detection.

Furthermore, my exploration of hyperparameters revealed valuable insights into model optimization. I discovered that a rank of 64 and a cosine learning rate scheduler for the SFTT trainer yielded slightly superior performance, emphasizing the critical role of hyperparameter tuning in maximizing model efficacy.

Moreover, my analysis of data preprocessing techniques highlighted the nuanced impact of stopword removal on model accuracy. Contrary to conventional wisdom, we observed a minor decrease in performance upon removing stopwords, most probably because of small length leading us to retain them in our preprocessing pipeline.

Chapter 8

Future work:

In future work, I aspire to explore additional parameter-efficient fine-tuning techniques like prefix tuning to optimize model performance while minimizing computational resources. Furthermore, I aim to delve into advanced methodologies such as Reinforcement Learning with Human Feedback (RLHF) for my Master's Thesis Project under Prof. PK Mishka's guidance. RLHF offers the potential to integrate human feedback into the training process, facilitating real-time adaptation based on annotator input. By leveraging RLHF, I anticipate achieving enhanced accuracy and robustness in identifying and mitigating gender biases and stereotypes within textual data. Through these future endeavors, I seek to contribute to the ongoing advancement of text classification techniques, fostering inclusivity and equity in AI applications.

References

1. Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

2. Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2024). Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, *36*.

3 Li, X. L., & Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

4. Rajbhandari, S., Rasley, J., Ruwase, O., & He, Y. (2020, November). Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis* (pp. 1-16). IEEE

5. Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

6.. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

7. Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., ... & Kenealy, K. (2024). Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.