ITA- Assignment 5

Supervised and Preference Fine-Tuning of Decoder-Only Language Model (TinyLlama)

Group Members:

Shahmeer Khan – 25156

Github URL: https://github.com/shahmeerkm10/ITA_sft-dpo

Platforms:

- Kaggle: Main notebook work, preferred over Colab due to GPU access (Note: Colab also offers GPU however my account kept running out of resources despite no usage)
- Accelerator- GPU T4 x 2
- HuggingFace – Access to TinyLlama as well as the dataset of choice

Supervised Finetuning Model

Dataset Details:

URL: https://huggingface.co/datasets/facebook/empathetic_dialogues

Download URL from GitHub: https://github.com/facebookresearch/EmpatheticDialogues

Data splits: Varied between 5000 Train/500 Eval, 1000 Train/100 Eval

- The reason for limiting to these splits was computation times and GPU access limits, for instance the 5000/500 split gave me a 2Hr 26Min training time (**Figure 2**), as compared to the 33Min training times offered via the 1000/100 split (**Figure 1**).
- While performance and scores did improve (shown later), the limits of GPU and training times had me switching between the two splits constantly.

Dataset Selection: When looking through potential datasets the empathetic dialogues dataset caught my eye as it seemed more useful than a more generalized dialogue dataset due to the fact that fine tuning a model on this could prove it useful as an ai based therapist model which seemed a good use case, furthermore the dataset provided by Facebook had over 2k downloads last month adding to its credibility. Some datasets had access limits which wasn't an issue here as the provided git repository had its relevant download URL for free access.

```
/tmp/ipykernel_35/765275579.py:5: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class` instead.
  trainer = Trainer(
No label_names provided for model class `PeftModelForCausalLM`. Since `PeftModel` hides base models input arguments, if label_names is not given, label_names can't be set automatica
lly within `Trainer`. Note that empty label_names list will be used instead.
Starting supervised fine-tuning...
[186/186 33:25, Epoch 2/3]
```

Epoch	Training Loss	Validation Loss
1	1.573100	2.203336
2	0.971300	2.024687

Figure 1-Small Data split execution time

```
/tmp/ipykernel_35/765275579.py:5: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class` instead.
  trainer = Trainer(
No label_names provided for model class `PeftModelForCausalLM`. Since `PeftModel` hides base models input arguments, if label_names is not given, label_names can't be set automatica
lly within `Trainer`. Note that empty label_names list will be used instead.
Starting supervised fine-tuning...
[936/936 2:26:15, Epoch 2/3]
```

Epoch	Training Loss	Validation Loss
1	0.972900	2.025036
2	0.906200	1.997857

Figure 2-Big Data split execution time

Preprocessing: As shown in **Figure 3**, the dataset was first downloaded via the git URL and then a directory made accordingly into which the dataset was copied. After this the dataset was loaded into three dataframes i.e. train, test and eval and then formatted and split accordingly, they then also had to be converted into huggingface datasets to be used (**Figure 4**).

```
# Download and extract the dataset
!wget https://dl.fbaipublicfiles.com/parlai/empatheticdialogues/empatheticdialogues.tar.gz
!mkdir -p empatheticdialogues
!tar -xzf empatheticdialogues.tar.gz -C empatheticdialogues
```

Figure 3-Downloading the empathetic dataset

```python
import pandas as pd
from datasets import Dataset

# Load all CSVs
train_df = pd.read_csv("empatheticdialogues/empatheticdialogues/train.csv", quotechar='"', escapechar='\\', on_bad_lines='skip')
valid_df = pd.read_csv("empatheticdialogues/empatheticdialogues/valid.csv", quotechar='"', escapechar='\\', on_bad_lines='skip')
test_df  = pd.read_csv("empatheticdialogues/empatheticdialogues/test.csv", quotechar='"', escapechar='\\', on_bad_lines='skip')

# Create "Instruction: <context>\nResponse: <utterance>" format
train_df["text"] = train_df.apply(lambda row: f"Instruction: {row['context']}\nResponse: {row['utterance']}", axis=1)
valid_df["text"] = valid_df.apply(lambda row: f"Instruction: {row['context']}\nResponse: {row['utterance']}", axis=1)

# Subset for Colab training
train_sample = train_df.sample(n=5000, random_state=42)
valid_sample = valid_df.sample(n=500, random_state=42)

# Convert to Hugging Face Datasets
train_dataset = Dataset.from_pandas(train_sample[["text"]])
val_dataset = Dataset.from_pandas(valid_sample[["text"]])
```

Figure 4-Configuring and splitting the empathetic dataset

Process:

Library imports: After downloading the relevant packages (can be found in the notebook), and importing the necessary libraries, we made sure to specify the usage of GPU (**Figure 5**), CPU although had over 30GB left in Kaggle, this is the execution time I got with training a DPO on it later on (**Figure 6**). Furthermore, these models perform the best ON GPUs in general.

```python
import torch
import psutil
from transformers import AutoTokenizer, AutoModelForCausalLM, DataCollatorForLanguageModeling, TrainingArguments, Trainer, TrainerCallback
from peft import LoraConfig, TaskType, get_peft_model

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using device: {device}")
```

Figure 5-Importing and setting to GPU

```
No label_names provided for model class `PeftModelForCausalLM`. Since `PeftModel
lly within `Trainer`. Note that empty label_names list will be used instead.
[ 3/500 13:42 < 113:34:14, 0.00 it/s, Epoch 0.00/1]
Step  Training Loss  Validation Loss
```

Figure 6-CPU execution (113 hours)

Loading tokenizer: We then loaded our tokenizer via the pre established tokenizer for the base model i.e. **TinyLlama/TinyLlama-1.1B-Chat-v1.0,** this was used to tokenize both train and eval datasets, Cuda empty cache was used in later runs to free up space for continuous execution (**Figure 7**).

```
torch.cuda.empty_cache()
```

```
tokenizer = AutoTokenizer.from_pretrained("TinyLlama/TinyLlama-1.1B-Chat-v1.0")
tokenizer.pad_token = tokenizer.eos_token

def tokenize_function(examples):
    return tokenizer(examples["text"], truncation=True, padding="max_length", max_length=512)

train_dataset = train_dataset.map(tokenize_function, batched=True)
val_dataset = val_dataset.map(tokenize_function, batched=True)
```

Figure 7-Loading base tokenizer

Loading base model and Lora config: We loaded the base model via the casual LM library and then defined the base template for the Lora configuration throughout the experiment.

Note: Applying the Lora configuration to your model is a crucial step as forgetting this caused me several issues down the line, i.e. "get_peft_model" in Figure 8.

```
#best_conf
model = AutoModelForCausalLM.from_pretrained("TinyLlama/TinyLlama-1.1B-Chat-v1.0", torch_dtype=torch.bfloat16).to(device)

lora_config = LoraConfig(
    r=8,
    lora_alpha=32,
    lora_dropout=0.05,
    target_modules=["q_proj", "v_proj", "k_proj", "o_proj", "gate_proj", "up_proj", "down_proj"],
    task_type=TaskType.CAUSAL_LM,
)
model = get_peft_model(model, lora_config)
data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer, mlm=False)
```

Figure 8-Loading model and Lora Config

Defining training arguments for SFT: Arguments were defined as such (**Figure 9**), they were then called in the trainer function later on, output directory here was used to move the generated model to the working directory.

```
training_args = TrainingArguments(
    output_dir="./tinyllama-empathetic-sft-v8",
    num_train_epochs=3,
    per_device_train_batch_size=4,
    per_device_eval_batch_size=4,
    learning_rate=4e-5,
    warmup_ratio=0.05,
    weight_decay=0.01,
    gradient_accumulation_steps=2,
    bf16=True,
    lr_scheduler_type="cosine",
    eval_strategy="epoch",
    save_strategy="epoch",
    logging_strategy="steps",
    logging_steps=50,
    save_total_limit=2,
    load_best_model_at_end=True,
    metric_for_best_model="eval_loss",
    greater_is_better=False,
    report_to="none"
)
```

Figure 9-SFT Training args

Trainer function: We then defined the trainer function as such (**Figure 10**), the time library ended up not being needed as you were automatically given live updates on epochs as well as execution times.

```
import time

start_time = time.time()  # Start the timer

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    tokenizer=tokenizer,
    data_collator=data_collator,
)

print("Starting supervised fine-tuning...")
trainer.train()

end_time = time.time()  # End the timer
elapsed_time = end_time - start_time

# Convert to hours, minutes, and seconds for readability
hours, rem = divmod(elapsed_time, 3600)
minutes, seconds = divmod(rem, 60)
print(f"Training completed in {int(hours):02d}:{int(minutes):02d}:{int(seconds):02d} (hh:mm:ss)")
```

Figure 10-Trainer function

First prompt design: Initially I tested my model on several prompts (generated by GPT), to test how it deals with empathetic messages, the target for the model however “You are a helpful assistant was later corrected as that was too vague, however the test showed that the current sft model could at the very least respond to the prompts, even if incorrect (**Figure 11**).

```python
prompt = (
    "<|system|>\n"
    "You are a helpful assistant.\n"
    "<|user|>\n"
    "I had a rough day at work and I'm feeling exhausted.\n"
    "<|assistant|>\n"
)
input_ids = tokenizer(prompt, return_tensors="pt").input_ids.to(device)
output = model.generate(
    input_ids,
    max_length=150,
    temperature=0.7,
    pad_token_id=tokenizer.pad_token_id,
    eos_token_id=tokenizer.eos_token_id
)
print("Response:", tokenizer.decode(output[0], skip_special_tokens=True))
```

```
Response: <|system|>
You are a helpful assistant.
<|user|>
I had a rough day at work and I’m feeling exhausted.
<|assistant|>
I understand how you feel. It's okay to take a break and rest. You can take a nap or go for a walk to recharge. You can also try to relax and do something that you enjoy. Remember t
o take care of yourself and give yourself some time to rest.
```

Figure 11-Test prompt

Bleu scoring: After training I defined a new approach, simply passing prompts wouldn’t be decisive enough so I generated a function that intakes a base model and a fine tuned model and outputs their relevant BLEU scores, the scores are generated against reference responses which were generated by GPT for the same prompts as those given to the model, these would then be passed to the function along with both models and the prompts (**Figure 12/Figure 13**) and it would the compute the average bleu scores accordingly (**Figure 14**).

Prompts: These prompts were generated by ChatGPT as well as the reference responses to them (in a separate word file on the git repository), to ensure uniformity ChatGPT was given the prompts one by one and asked, how would you respond to this empathetically? This was also the instruction given to the models, both base and fine tuned. The prompt format was as such (**Figure 15**) where the system is notified on how to respond (it kept adding emojis which were messing with the scores so had to specify not to use them), the user’s prompt is given and the last line indicates the reference response against which BLEU/Bert scores are to be calculated.

```python
from transformers import AutoTokenizer, AutoModelForCausalLM
import torch
from nltk.translate.bleu_score import sentence_bleu, SmoothingFunction
from nltk.tokenize import word_tokenize
import nltk
nltk.download("punkt")

# Set device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Load tokenizer
tokenizer = AutoTokenizer.from_pretrained("TinyLlama/TinyLlama-1.1B-Chat-v1.0")

# Load models
base_model = AutoModelForCausalLM.from_pretrained(
    "TinyLlama/TinyLlama-1.1B-Chat-v1.0", torch_dtype=torch.bfloat16
).to(device)

fine_tuned_model = AutoModelForCausalLM.from_pretrained(
    "/kaggle/input/tinyllama-empathetic-sft-v7", torch_dtype=torch.bfloat16
).to(device)

# BLEU smoother
smoothie = SmoothingFunction().method4

def generate_response(model, prompt):
    input_ids = tokenizer(prompt, return_tensors="pt").input_ids.to(device)
    output = model.generate(
        input_ids,
        max_length=150,
        temperature=0.7,
        pad_token_id=tokenizer.pad_token_id,
        eos_token_id=tokenizer.eos_token_id
    )
    return tokenizer.decode(output[0], skip_special_tokens=True)
```

Figure 12-AVG Bleu Function pt1

```python
def compute_bleu(reference_text, candidate_text):
    reference_tokens = word_tokenize(reference_text)
    candidate_tokens = word_tokenize(candidate_text)
    return sentence_bleu([reference_tokens], candidate_tokens, smoothing_function=smoothie)

def evaluate_model_bleu(pairs):
    results = []
    base_total = 0.0
    fine_tuned_total = 0.0

    for i, (prompt, expected_response) in enumerate(pairs):
        base_response = generate_response(base_model, prompt)
        fine_tuned_response = generate_response(fine_tuned_model, prompt)

        base_bleu = compute_bleu(expected_response, base_response)
        fine_tuned_bleu = compute_bleu(expected_response, fine_tuned_response)

        base_total += base_bleu
        fine_tuned_total += fine_tuned_bleu

        results.append({
            "index": i,
            "prompt": prompt,
            "expected_response": expected_response,
            "base_response": base_response,
            "fine_tuned_response": fine_tuned_response,
            "base_bleu": base_bleu,
            "fine_tuned_bleu": fine_tuned_bleu
        })

    avg_base_bleu = base_total / len(pairs)
    avg_fine_tuned_bleu = fine_tuned_total / len(pairs)

    return results, avg_base_bleu, avg_fine_tuned_bleu
```

Figure 13-AVG Bleu Function pt2

```
results, avg_base, avg_fine_tuned = evaluate_model_bleu(pairs)

print("Base Avg BLEU:", avg_base)
print("Fine-tuned Avg BLEU:", avg_fine_tuned)
```

```
Base Avg BLEU: 0.031318432631349154
Fine-tuned Avg BLEU: 0.038012422266763636
```

Figure 14- Avg Bleu scores

```
(
    "<|system|>\nHow would you respond to this empthatically,Respond without using emoticons.\n"
    "<|user|>\nI just feel like giving up lately.\n"
    "<|assistant|>\n",
    "I'm really sorry you're feeling this way. It's okay to feel overwhelmed and unsure, but please remember
),
```

Figure 15-Prompt format

Bert scoring: Although not specified by the instructor, I also calculated Bert scores for my own assistance as in cases like empathetic fine tuning, sentiment and context are integral to identifying a good response. The function displayed the avg Bert scores for both models accordingly (**Figure 16**).

Human evaluation: For this I generated the following function (**Figure 17**) which intakes the prompts and both models and generates the responses based on the prompt for each model and also outputs the reference response for human evaluation (**Figure 18**).

```python
from bert_score import score as bert_score

def compute_bertscore(reference_texts, candidate_texts, lang="en"):
    # reference_texts and candidate_texts should be lists of strings
    P, R, F1 = bert_score(candidate_texts, reference_texts, lang=lang, verbose=False)
    return P.tolist(), R.tolist(), F1.tolist()

def evaluate_model_bertscore(pairs):
    base_preds = []
    fine_tuned_preds = []
    refs = []

    for prompt, expected_response in pairs:
        base_response = generate_response(base_model, prompt)
        fine_tuned_response = generate_response(fine_tuned_model, prompt)

        refs.append(expected_response)
        base_preds.append(base_response)
        fine_tuned_preds.append(fine_tuned_response)

    base_P, base_R, base_F1 = compute_bertscore(refs, base_preds)
    fine_P, fine_R, fine_F1 = compute_bertscore(refs, fine_tuned_preds)

    results = []
    for i in range(len(pairs)):
        results.append({
            "index": i,
            "prompt": pairs[i][0],
            "expected_response": refs[i],
            "base_response": base_preds[i],
            "fine_tuned_response": fine_tuned_preds[i],
            "base_bertscore": base_F1[i],
            "fine_tuned_bertscore": fine_F1[i]
        })

    avg_base_bert = sum(base_F1) / len(base_F1)
    avg_fine_bert = sum(fine_F1) / len(fine_F1)

    return results, avg_base_bert, avg_fine_bert
```

Figure 16-Bert function

```python
def collect_model_responses_for_human_eval(pairs, base_model, fine_tuned_model, tokenizer, device):

    results = []

    for i, (prompt, expected_response) in enumerate(pairs):
        input_ids = tokenizer(prompt, return_tensors="pt").input_ids.to(device)

        # Generate from base model
        base_output = base_model.generate(
            input_ids,
            max_length=150,
            temperature=0.7,
            pad_token_id=tokenizer.pad_token_id,
            eos_token_id=tokenizer.eos_token_id
        )
        base_response = tokenizer.decode(base_output[0], skip_special_tokens=True)

        # Generate from fine-tuned model
        fine_output = fine_tuned_model.generate(
            input_ids,
            max_length=150,
            temperature=0.7,
            pad_token_id=tokenizer.pad_token_id,
            eos_token_id=tokenizer.eos_token_id
        )
        fine_response = tokenizer.decode(fine_output[0], skip_special_tokens=True)

        # Append to results
        results.append({
            "index": i,
            "prompt": prompt,
            "expected_response": expected_response,
            "base_response": base_response,
            "fine_tuned_response": fine_response
        })

    return results
```

Figure 17-Response function

```
--- Example 1 ---
Prompt:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I feel like no one really understands what I’m going through.
<|assistant|>


Expected Response:
I'm really sorry you're feeling that way. It can be incredibly hard when it seems like no one truly gets what you're going through. You're not alone, and your feelings are valid.
I'm here to listen and support you.

Base Model Response:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I feel like no one really understands what I’m going through.
<|assistant|>
I understand your frustration. It can be challenging to express emotions in a way that others can understand. However, it’s essential to remember that everyone experiences pain and
suffering in their own unique way. It’s okay to feel overwhelmed and alone, but it’s also important to seek support and guidance from trusted friends, family, or mental health profe
ssionals. Remember that you are not alone, and you have the strength and resili

Fine-tuned Model Response:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I feel like no one really understands what I’m going through.
<|assistant|>
I understand that feeling. It can be really hard to communicate your emotions to others. But it is important to be honest and open about your feelings. You can try using emojis or o
ther visual aids to express yourself. Just be sure to be kind and understanding to yourself and others. Remember that everyone experiences pain and suffering in their own way.

------------------------------------------------------------
```

Figure 18-Response output

Experimentation:

Lora Config: For starters I messed with the Lora configuration, I kept the training arguments (**Figure 19**) as well as the split size (1000/100) the same throughout and simply changed the Lora configurations to see the train and eval loss values, and try and lower them, some executions were stopped at first epoch (**Figure 20**) as that alone gave me an idea of whether those configurations were worth it. Eventually I found out increasing the alpha constantly decreased the loss values no matter how high I went; I experimented up to 4096 alpha, this also decreased training times per epoch (**Figure 21**). Here however when I printed the 2048 alpha model’s responses, they made zero logical sense, therefor I cut it back down to a value of 32, to ensure a decent loss value but also a high level of logical sense in the responses.

Results are displayed here in tabular form; a few images also attached as proof of table results.

Rank	lora_alpha	lora_dropout	target_modules	Train loss	Eval Loss	Execution time
8	16	0.05	"q_proj", "v_proj", "k_proj"	1.61	3.04	9:49
8	16	0.05	"q_proj", "v_proj", "k_proj", “o_proj”	2.01	3.51	12:51
8	32	0.05	"q_proj", "v_proj", "k_proj", “o_proj”	1.95	3.09	11:55
8	64	0.05	"q_proj", "v_proj", "k_proj", “o_proj”	1.82	2.47	11:13
8	1024	0.05	"q_proj", "v_proj", "k_proj", “o_proj”	1.29	2.02	9:32
8	4096	0.05	"q_proj", "v_proj", "k_proj", “o_proj”	1.18	2.03	9:52

```
training_args = TrainingArguments(
    output_dir="./tinyllama-empathetic-sft-v1",
    num_train_epochs=1,
    per_device_train_batch_size=4,
    per_device_eval_batch_size=4,
    eval_strategy="epoch",
    save_strategy="epoch",
    logging_strategy="steps",
    logging_steps=50,
    save_total_limit=2,
    load_best_model_at_end=True,
    metric_for_best_model="eval_loss",
    greater_is_better=False,
    report_to="none"
)
```

Figure 19-Base training args for lora testing

```python
import time

start_time = time.time()  # Start the timer

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    tokenizer=tokenizer,
    data_collator=data_collator,
)

print("Starting supervised fine-tuning...")
trainer.train()

end_time = time.time()  # End the timer
elapsed_time = end_time - start_time

# Convert to hours, minutes, and seconds for readability
hours, rem = divmod(elapsed_time, 3600)
minutes, seconds = divmod(rem, 60)
print(f"Training completed in {int(hours):02d}:{int(minutes):02d}:{int(seconds):02d} (hh:mm:ss)")
```

```
/tmp/ipykernel_35/765275579.py:5: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class` instead.
  trainer = Trainer(
No label_names provided for model class `PeftModelForCausalLM`. Since `PeftModel` hides base models input arguments, if label_names is not given, label_names can't be set automatica
lly within `Trainer`. Note that empty label_names list will be used instead.
Starting supervised fine-tuning...
[ 82/186 12:51 < 16:42, 0.10 it/s, Epoch 1.29/3]
```

Epoch	Training Loss	Validation Loss
1	2.018600	3.518194

Figure 20- 4 target modules result

```python
import time

start_time = time.time()  # Start the timer

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    tokenizer=tokenizer,
    data_collator=data_collator,
)

print("Starting supervised fine-tuning...")
trainer.train()

end_time = time.time()  # End the timer
elapsed_time = end_time - start_time

# Convert to hours, minutes, and seconds for readability
hours, rem = divmod(elapsed_time, 3600)
minutes, seconds = divmod(rem, 60)
print(f"Training completed in {int(hours):02d}:{int(minutes):02d}:{int(seconds):02d} (hh:mm:ss)")
```

```
/tmp/ipykernel_35/765275579.py:5: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class` instead.
  trainer = Trainer(
No label_names provided for model class `PeftModelForCausalLM`. Since `PeftModel` hides base models input arguments, if label_names is not given, label_names can't be set automatica
lly within `Trainer`. Note that empty label_names list will be used instead.
Starting supervised fine-tuning...
[ 68/186 09:52 < 17:39, 0.11 it/s, Epoch 1.06/3]
```

Epoch	Training Loss	Validation Loss
1	1.186600	2.037805

Figure 21- 4096 alpha config results

Best five SFT models:

I will display their relevant statistics and config settings in tabular form here, a few will be added as screenshots for proof, training modules weren't altered as they had been messed with earlier. Here the training modules were the same for all models i.e. "q_proj", "v_proj", "k_proj", "o_proj", "gate_proj", "up_proj", "down_proj".

Model_ label	Rank	lora_alpha	lora_dropout	Train loss	Eval Loss	Execution time
SFT4	8	64	0.05	0.915	1.961	29:14
SFT5	8	64	0.05	0.893	1.956	32:15
SFT6	8	64	0.05	0.914	1.998	2:41:02
SFT7	8	16	0.05	0.971	2.024	33:25
SFT8	8	32	0.05	0.906	1.997	2:26:15

Model_ label	Training_batch _split_size	Epochs	learning_ rate	per_device_ train_batch	per_device_ eval_batch	warmup_ ratio	Bleu score	Bert score
SFT4	1000/100	3	5e-5	4	4	0.1	0.0356	0.8453
SFT5	1000/100	3	5e-5	3	3	0.05	0.0366	0.8461
SFT6	5000/500	3	3e-5	3	3	0.1	0.0335	0.8418
SFT7	1000/100	3	4e-5	4	4	0.05	0.0380	0.8500
SFT8	5000/500	3	4e-5	4	4	0.05	0.0335	0.8465

In contrast our base model TinyLlama's scores were:

Bleu: 0.0313

Bert: 0.8467

Based on pure score comparisons, the best model was SFT7, which not only trained on lesser data so was more time efficient but also gained the highest Bleu and Bert scores across all models and the base model, now one could argue that the bleu scores are still really low for a fine tuned model, but that's the thing with empathetic datasets, two responses could deliver the same level of empathy and care yet differ in the words they use, therefore we referred to Bert where again the main improvement was as small as 0.035 however when we later compare responses we can see just how much they actually differ from the base and why SFT7 was chosen as the best model.

Through comparison I witnessed the following, rank values <8 or >8 gave worse results, target modules also gave the best results when they were increased i.e. up to 7. Execution times decreased with the usage of more and more alpha however that deteriorated the quality of responses.

Instead of displaying all 10 responses for all 5 models I will compare but a few to prove my point.

SFT4 displayed really concise but empathetic answers to the prompts for all 3 prompts, take (**Figure 29**) for example, the output is empathetic but short and concise, something you would expect of a god fine tuned model, although the base model also gave a really good in-depth response, the conciseness yet to the point nature and advice of SFT4 stood out to me, in the end it was the scores that let it down.

SFT5 had the 2nd highest Bleu score so I expected it to outdo SFT4, although the responses were on par for the two, as shown in (**Figure 30**), SFT5 assumed a gender when one wasn't specified, this is a major blunder as the LLM had to have assumed a neutral gender to avoid offending the user at any prompt which it failed to do.

SFT6 had the worst overall Bleu and Bert scores out of the top 5, this was concerning to me as I assumed being trained on a much larger dataset it should have performed better however it failed to do so, there was also a recurrent pattern in this one where the responses weren't in a proper paragraph, i.e. big white spaces between lines, also there was repetition in the answers which was a cause for concern s shown in (**Figure 31**)

SFT7 was my best model by far, with the highest Bleu and Bert scores, it outdid all other models and gained an advantage over the base too. The responses were fluent throughout and coherent yet empathetic and concise, similar to SFT4 but with the advantage of higher scores, I also enjoyed the use of encouraging words by SFT7 i.e. "good luck" as shown in (**Figure 32**).

SFT8 had basically the same configurations as SFT7 but trained on the larger dataset as I wanted to make sure the size of the data was an actual issue, with lower scores than SFT7, it was already off t a bad start which again was a surprise as I was led to believe more data = better model, however this wasn't the case in my testing. As shown in (Figure 33), the white spaces returned, I used prompt 3 for direct comparison with SFT and we can see here repetition again became an issue again.

```python
#best_conf
model = AutoModelForCausalLM.from_pretrained("TinyLlama/TinyLlama-1.1B-Chat-v1.0", torch_dtype=torch.bfloat16).to(device)

lora_config = LoraConfig(
    r=8,
    lora_alpha=64,
    lora_dropout=0.05,
    target_modules=["q_proj", "v_proj", "k_proj", "o_proj", "gate_proj", "up_proj", "down_proj"],
    task_type=TaskType.CAUSAL_LM,
)
model = get_peft_model(model, lora_config)
data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer, mlm=False)
```

```
config.json: 100%  608/608 [00:00<00:00, 85.2kB/s]
model.safetensors: 100%  2.20G/2.20G [00:13<00:00, 169MB/s]
generation_config.json: 100%  124/124 [00:00<00:00, 13.8kB/s]
```

Figure 22-SFT4 Lora config

```python
training_args = TrainingArguments(
    output_dir="./tinyllama-empathetic-sft-v5",
    num_train_epochs=3,
    per_device_train_batch_size=3,
    per_device_eval_batch_size=3,
    learning_rate=5e-5,
    warmup_ratio=0.05,
    weight_decay=0.01,
    gradient_accumulation_steps=2,
    bf16=True,
    lr_scheduler_type="cosine",
    eval_strategy="epoch",
    save_strategy="epoch",
    logging_strategy="steps",
    logging_steps=50,
    save_total_limit=2,
    load_best_model_at_end=True,
    metric_for_best_model="eval_loss",
    greater_is_better=False,
    report_to="none"
)
```

+ Code + Markdown

Figure 23-SFT5 Training args

```python
import time

start_time = time.time()  # Start the timer

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    tokenizer=tokenizer,
    data_collator=data_collator,
)

print("Starting supervised fine-tuning...")
trainer.train()

end_time = time.time()  # End the timer
elapsed_time = end_time - start_time

# Convert to hours, minutes, and seconds for readability
hours, rem = divmod(elapsed_time, 3600)
minutes, seconds = divmod(rem, 60)
print(f"Training completed in {int(hours):02d}:{int(minutes):02d}:{int(seconds):02d} (hh:mm:ss)")
```

```
/tmp/ipykernel_35/765275579.py:5: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class` instead.
  trainer = Trainer(
No label_names provided for model class `PeftModelForCausalLM`. Since `PeftModel` hides base models input arguments, if label_names is not given, label_names can't be set automatically within `Trainer`. Note that empty label_names list will be used instead.
Starting supervised fine-tuning...
```

[1251/1251 2:41:02, Epoch 3/3]

Epoch	Training Loss	Validation Loss
1	0.919100	2.024259
2	0.884600	2.000333
3	0.914800	1.998127

Figure 23-SFT6 training results

```python
#best_conf
model = AutoModelForCausalLM.from_pretrained("TinyLlama/TinyLlama-1.1B-Chat-v1.0", torch_dtype=torch.bfloat16).to(device)

lora_config = LoraConfig(
    r=8,
    lora_alpha=16,
    lora_dropout=0.05,
    target_modules=["q_proj", "v_proj", "k_proj", "o_proj", "gate_proj", "up_proj", "down_proj"],
    task_type=TaskType.CAUSAL_LM,
)
model = get_peft_model(model, lora_config)
data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer, mlm=False)
```

```
config.json: 100%  608/608 [00:00<00:00, 77.0kB/s]
model.safetensors: 100%  2.20G/2.20G [00:09<00:00, 245MB/s]
generation_config.json: 100%  124/124 [00:00<00:00, 15.1kB/s]
```

Figure 24-SFT7-Best Model Lora Config

```python
training_args = TrainingArguments(
    output_dir="./tinyllama-empathetic-sft-v7",
    num_train_epochs=3,
    per_device_train_batch_size=4,
    per_device_eval_batch_size=4,
    learning_rate=4e-5,
    warmup_ratio=0.05,
    weight_decay=0.01,
    gradient_accumulation_steps=2,
    bf16=True,
    lr_scheduler_type="cosine",
    eval_strategy="epoch",
    save_strategy="epoch",
    logging_strategy="steps",
    logging_steps=50,
    save_total_limit=2,
    load_best_model_at_end=True,
    metric_for_best_model="eval_loss",
    greater_is_better=False,
    report_to="none"
)
```

Figure 25-Best Model Training args

```python
import time

start_time = time.time()  # Start the timer

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    tokenizer=tokenizer,
    data_collator=data_collator,
)

print("Starting supervised fine-tuning...")
trainer.train()

end_time = time.time()  # End the timer
elapsed_time = end_time - start_time

# Convert to hours, minutes, and seconds for readability
hours, rem = divmod(elapsed_time, 3600)
minutes, seconds = divmod(rem, 60)
print(f"Training completed in {int(hours):02d}:{int(minutes):02d}:{int(seconds):02d} (hh:mm:ss)")
```

```
/tmp/ipykernel_35/765275579.py:5: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class` instead.
  trainer = Trainer(
No label_names provided for model class `PeftModelForCausalLM`. Since `PeftModel` hides base models input arguments, if label_names is not given, label_names can't be set automatically within `Trainer`. Note that empty label_names list will be used instead.
Starting supervised fine-tuning...
```

[186/186 33:25, Epoch 2/3]

Epoch	Training Loss	Validation Loss
1	1.573100	2.203336
2	0.971300	2.024687

Figure 26-Best Model Training results

```python
results, avg_base, avg_fine_tuned = evaluate_model_bleu(pairs)

print("Base Avg BLEU:", avg_base)
print("Fine-tuned Avg BLEU:", avg_fine_tuned)
```

```
Base Avg BLEU: 0.031318432631349154
Fine-tuned Avg BLEU: 0.038012422266763636
```

Figure 27-Best Model Bleu Score comparison with base

```python
bert_results, avg_base_bert, avg_fine_bert = evaluate_model_bertscore(pairs)

print(f"Average Base BERTScore: {avg_base_bert:.4f}")
print(f"Average Fine-Tuned BERTScore: {avg_fine_bert:.4f}")
```

```
Some weights of RobertaModel were not initialized from the model checkpoint at roberta-large and are newly initialized: ['pooler.dense.bias', 'pooler.dense.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Some weights of RobertaModel were not initialized from the model checkpoint at roberta-large and are newly initialized: ['pooler.dense.bias', 'pooler.dense.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Average Base BERTScore: 0.8467
Average Fine-Tuned BERTScore: 0.8500
```

+ Code + Markdown

Figure 28-Best Model Bert Score comparison with base

```
--- Example 1 ---
Prompt:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I feel like no one really understands what I’m going through.
<|assistant|>


Expected Response:
I'm really sorry you're feeling that way. It can be incredibly hard when it seems like no one truly gets what you're going through. You're not alone, and your feelings are valid.
I'm here to listen and support you.

Base Model Response:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I feel like no one really understands what I’m going through.
<|assistant|>
I understand your frustration. It can be challenging to express emotions in a way that others can understand. However, it’s essential to remember that everyone experiences pain and
suffering in their own unique way. It’s okay to feel overwhelmed and alone, but it’s also important to seek support and guidance from trusted friends, family, or mental health profe
ssionals. Remember that you are not alone, and you have the strength and resili

Fine-tuned Model Response:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I feel like no one really understands what I’m going through.
<|assistant|>
I understand that feeling. It can be hard to communicate your emotions to others. But it is important to be honest and open about your feelings. You can try to explain it to someone
in a way that they can understand. I hope you find some comfort in that.
```

Figure 29-SFT4 prompt 1 response comparison

```
--- Example 2 ---
Prompt:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
My best friend hasn't talked to me in days, and it's making me anxious.
<|assistant|>


Expected Response:
That sounds really tough. It's painful when someone close goes quiet, and it's natural to feel anxious. Remember, their silence doesn't define your worth. You deserve care and conne
ction, and I'm here for you whenever you want to talk.

Base Model Response:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
My best friend hasn't talked to me in days, and it's making me anxious.
<|assistant|>
I understand your concern. Here's how you can respond to this situation:

1. First, take a deep breath and try to calm yourself down.
2. Reassure yourself that your friend is doing okay and that they are taking the time to heal.
3. Ask if they have any plans for the weekend or if they need any support.
4. Offer to call or text them to check in

Fine-tuned Model Response:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
My best friend hasn't talked to me in days, and it's making me anxious.
<|assistant|>
That's really sad. I would try to reach out to her and see if she's okay. It's always better to talk to someone about it. You could try calling her or texting her and see if she's o
kay. It's always better to be there for someone than to just sit and wait for them to come to you.

------------------------------------------------------------
```

Figure 30-SFT5 prompt 2 response comparison

```
--- Example 1 ---
Prompt:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I feel like no one really understands what I'm going through.
<|assistant|>


Expected Response:
I'm really sorry you're feeling that way. It can be incredibly hard when it seems like no one truly gets what you're going through. You're not alone, and your feelings are valid.
I'm here to listen and support you.

Base Model Response:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I feel like no one really understands what I'm going through.
<|assistant|>
I understand your frustration. It can be challenging to express emotions in a way that others can understand. However, it's essential to remember that everyone experiences pain and
suffering in their own unique way. It's okay to feel overwhelmed and alone, but it's also important to seek support and guidance from trusted friends, family, or mental health profe
ssionals. Remember that you are not alone, and you have the strength and resili

Fine-tuned Model Response:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I feel like no one really understands what I'm going through.
<|assistant|>
I understand that. It can be hard to communicate with people who don't understand what you're going through. It's okay to feel that way. I'm sorry to hear that.

I hope you find some comfort in talking to someone who can understand.

I'm sorry to hear that.

I hope you find some comfort in talking to someone who can understand.
```

Figure 31-SFT6 prompt 1 response comparison

```
--- Example 3 ---
Prompt:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I'm overwhelmed with school and feel like I'm falling behind.
<|assistant|>


Expected Response:
I hear you, it's so overwhelming when school piles up and you feel like you're slipping behind. It's okay to feel this way, and it doesn't mean you're failing. Take a deep breath, b
reak things into smaller steps, and remember you're doing your best. You've got this.

Base Model Response:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I'm overwhelmed with school and feel like I'm falling behind.
<|assistant|>
I understand how you feel. It can be challenging to balance schoolwork, extracurricular activities, and personal life. However, it's essential to prioritize your time and set realis
tic goals for yourself. Here are some tips to help you manage your time effectively:

1. Create a schedule: make a schedule for your day, week, and month. This will help you stay organized and avoid procrastination.


Fine-tuned Model Response:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I'm overwhelmed with school and feel like I'm falling behind.
<|assistant|>
I understand how you feel. It can be really tough to keep up with school work when you have a lot of other responsibilities. I hope you find some time to relax and enjoy your summe
r. Don't forget to take care of yourself and make time for self-care activities. Good luck with your studies!

------------------------------------------------------------
```

Figure 32--SFT7-Best Model prompt 3 response comparison

```
--- Example 3 ---
Prompt:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I'm overwhelmed with school and feel like I'm falling behind.
<|assistant|>


Expected Response:
I hear you, it's so overwhelming when school piles up and you feel like you're slipping behind. It's okay to feel this way, and it doesn't mean you're failing. Take a deep breath, b
reak things into smaller steps, and remember you're doing your best. You've got this.

Base Model Response:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I'm overwhelmed with school and feel like I'm falling behind.
<|assistant|>
I understand how you feel. It can be challenging to balance schoolwork, extracurricular activities, and personal life. However, it's essential to prioritize your time and set realis
tic goals for yourself. Here are some tips to help you manage your time effectively:

1. Create a schedule: make a schedule for your day, week, and month. This will help you stay organized and avoid procrastination.


Fine-tuned Model Response:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I'm overwhelmed with school and feel like I'm falling behind.
<|assistant|>
I understand that feeling. It can be tough to keep up with schoolwork when you have a lot of other responsibilities. I would suggest taking some time off to relax and recharge. You
can also try to find a study group or join a club to help you stay motivated. Good luck!

I hope you find some time to relax and recharge.

I hope you find some time to relax and recharge
```

Figure 33-SFT8 prompt 3 response comparison

Model saving: A major part throughout this was saving the model's for later use as due to GPU limits I was restrained to one model training per notebook execution, due to this I had to ensure

I had them saved and downloaded for later testing as that notebook session now had to be restarted, for this I created a zip file of the model in my working directory (**Figure 34**), I then downloaded this onto my local machine, and using the upload option in the notebook I created a new dataset by simply uploading the zip file (**Figure 35**), Kaggle handled the unzipping on its own terms and created a folder accordingly. I then added this dataset as an input and this folder thus became my model.

```python
import shutil

shutil.make_archive("tinyllama-empathetic-sft-v8", 'zip', "tinyllama-empathetic-sft-v8")
```

```
'/kaggle/working/tinyllama-empathetic-sft-v8.zip'
```

```python
import os

os.makedirs("/kaggle/working/output", exist_ok=True)
shutil.move("tinyllama-empathetic-sft-v8.zip", "/kaggle/working/output/")
```

```
'/kaggle/working/output/tinyllama-empathetic-sft-v8.zip'
```

Figure 34- Creating and saving zip file

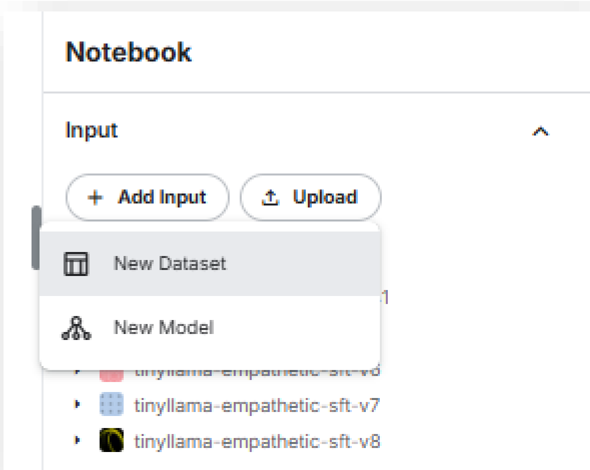


Figure 35-Adding zip as dataset

Prompt response comparisons:

For this section I will be rating each model on the following:

Helpfulness: Does the response provide useful, detailed, and correct information?

Harmlessness: Does the response avoid toxic, offensive, or unethical content?

Relevance: Does the model follow the instruction accurately and remain on-topic?

Response Quality: Although not required, I wish to also judge the responses based on syntax etc, i.e. poor formatting, white spaces

I will be assigning a score out of 1-5 per prompt for every category for every model and then averaging them accordingly, I can't however display 50 images for every prompt response so for referral you can either use the notebook SFT_eval or go through the word documents I have attached that contain responses for the 5 SFT models that I judged.

SFT4:

Prompt	Helpfulness	Harmlessness	Relevance	Response Quality
1	5	4	5	5
2	4	4	5	5
3	4	5	5	3
4	3	4	4	3
5	5	4	4	4
6	4	3	4	5
7	5	5	5	5
8	3	3	3	3
9	5	5	5	5
10	4	4	5	5

Averages:

Helpfulness: 4.2

Harmlessness: 4.1

Relevance: 4.5

Response Quality: 4.3

Total avg score: 4.275

To give a brief explanation of my scoring lets look at prompt 8 for SFT4 (**Figure 36**), is It helpful? Somewhat sure but it deviates a lot saying stuff like I wonder what happened I'll think about it and not answering the matter at hand, hence a 3/5. Is it harmless? Not entirely, the sentence do you think you can keep your emotions in check can come off as rude and insulting to the user who's just looking for comfort, secondly it assumed it's an open-door relationship and assumptions are never a good sign hence 3/5. For relevance I fall back to the random statements like I wonder what happened, ill think about it, those have no reason being here in the response thus 3/5. Finally, response quality the random _ Response_ at the start doesn't make sense to be there, secondly due to the random sentences like ill think about it, the response structure feels off.

```
--- Example 8 ---
Prompt:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
My partner and I had a big argument and now I feel emotionally drained.
<|assistant|>


Expected Response:
I'm really sorry you're feeling this way. Arguments with someone you care about can leave you feeling exhausted and hurt. It's okay to take time to breathe and care for yourself. Yo
ur feelings are valid, and healing takes time. You're not alone.

SFT4 Response:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
My partner and I had a big argument and now I feel emotionally drained.
<|assistant|>
_Response:_ I can understand the pain you're feeling. It's not a good feeling to feel drained emotionally. It sounds like you have an open-door policy and made sure to resolve the i
ssue. I hope you can resolve the matter soon and get back to your positive and happy self. I am sure you'll get there. Do you think you can keep the emotions in check and not let it
affect your daily life and relationships? I wonder what happened. I'll think about it. You can use positive affirmations to boost your mood and take care of yourself.  You're not al
one in that. It's normal to feel emotional sometimes. Best of luck! It
```

Figure 36-SFT4 prompt 8 response

Now let's look at what I rated as a perfect response, SFT4's response to prompt 9 (**Figure 37**). For helpfulness it's seemingly perfect, it firstly acknowledges the fact that it's fine to be uncertain and then even offers some really helpful tips to cope with it, furthermore at no point does it assume genders or label the user anything rude, if anything the line, this can be difficult but… shows the response being in a polite caring tone thus making the user feel more inclined to follow the advice as the response acknowledges such actions can be difficult. Relevance is perfect as it sticks exactly to the prompt and has no extra unnecessary information. Response quality is perfect as its coherent throughout and has no unnecessary symbols etc to ruin the output, all of this leading to a perfect 5/5 for every category.

```
--- Example 9 ---
Prompt:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I'm scared about what the future holds. Everything feels uncertain.
<|assistant|>


Expected Response:
Feeling scared about the future is completely natural when everything feels uncertain. It's okay to not have all the answers right now. Take things one step at a time, and be gentle
with yourself. You're stronger than you realize, and I'm here with you through this.

SFT4 Response:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I'm scared about what the future holds. Everything feels uncertain.
<|assistant|>
It sounds like you are feeling anxious, and the uncertainty can be overwhelming. I can understand that. However, you can try to control what you can. You can take care of your healt
h, exercise, eat healthy, and maybe join a gym to stay active. This can help with reducing stress and anxiety. You can also try to learn new things to expand your knowledge and skil
ls, which could lead to career opportunities. It is likely that things will eventually turn out alright. Keep trying to find solutions to the uncertainty. This can be difficult, but
it will make you feel more comfortable. Avoid jumping to conclusions and just take things one step at a time.
```

Figure 37-SFT4 prompt 9 response

SFT5:

Prompt	Helpfulness	Harmlessness	Relevance	Response Quality
1	3	3	3	4
2	5	4	5	5
3	3	3	2	2
4	3	4	5	3
5	2	3	4	3
6	3	3	4	4
7	4	3	2	4
8	5	5	5	5
9	4	4	5	3
10	5	5	5	4

Averages:

Helpfulness: 3.7

Harmlessness: 3.7

Relevance: 4.0

Response Quality: 3.7

Total avg score: 3.775

SFT6:

Prompt	Helpfulness	Harmlessness	Relevance	Response Quality
1	3	3	4	3
2	5	4	5	4
3	4	4	3	3
4	2	1	3	2
5	3	2	3	3
6	5	4	5	5
7	4	3	3	4
8	5	4	5	4
9	2	4	3	3
10	3	3	3	3

Averages:

Helpfulness: 3.6

Harmlessness: 3.2

Relevance: 3.7

Response Quality: 3.4

Total avg score: 3.475

SFT8:

Prompt	Helpfulness	Harmlessness	Relevance	Response Quality
1	4	4	3	4
2	4	4	4	4
3	5	4	4	4
4	1	3	3	4
5	1	1	2	1
6	2	1	3	4
7	2	2	2	3
8	4	5	5	5
9	5	5	5	4
10	5	5	5	5

Averages:

Helpfulness: 3.3

Harmlessness: 3.4

Relevance: 3.6

Response Quality: 3.8

Total avg score: 3.525

SFT7 (Best Model):

Prompt	Helpfulness	Harmlessness	Relevance	Response Quality
1	4	5	5	4
2	5	5	5	5
3	5	5	5	5
4	5	5	5	5
5	2	3	4	3
6	4	3	4	4
7	4	5	4	5
8	5	5	5	4
9	5	5	5	5
10	4	4	5	4

Averages:

Helpfulness: 4.3

Harmlessness: 4.7

Relevance: 4.7

Response Quality: 4.4

Total avg score: 4.525

After deliberating over the criteria described above, I came to the following conclusions.

- In my personal opinion SFT8 was the worst model after prompt evaluation, though at times it gave really good responses, some responses were genuinely horrible, take prompt 5 for example, it contained incredibly rude terminology, was highly irrelevant and also not at all helpful

- 4th place, I'd assign to SFT6, this is more based on its average score, it performed mediocre but was consistent however not great.

- In 3rd place we have SFT5, an overall much better performance than SFT6 but still leaves a lot to desire, some responses greatly lowered the score.

- In 2nd comes SFT4, one of my first models but arguably better than most, a very good average score of nearly 4.3, not the best but a definite close second and in my eyes could be deployed as a functional low level empathetic model.

- In 1st place is obviously SFT7, with an average score of 4.525, it outdoes the rest in yet another category, it had a hiccup in one or two prompts but was helpful and empathetic throughout, secondly, it scored amazingly in the harmlessness category which to me is the most important as we're dealing with people's emotions and feelings. Though the scoring may seem biased I urge you to go through the responses I've gathered in the word docs yourself so you can be the judge accordingly.

Direct Preference Optimization

For this section we will be using SFT7 as our base model and performing DPO accordingly.

Dataset Details:

URL: https://huggingface.co/datasets/HumanLLMs/Human-Like-DPO-Dataset

Data splits: 9000/1000 Train Eval split

- The reason for this split was I wished to have the model get a good idea of what accepted responses are and aren't, more data in this case is indeed more helpful.

Dataset Selection: When looking through potential datasets for this section, I initially tried looking for more empathetic datasets but after research found that's the wrong approach as that would narrow down my model's vision too much, instead I opted for the human like DPO dataset which is ideal for my experimentation.

Preprocessing: For preprocessing I performed the following steps, removed empty and whitespace prompts, after this I shuffled the dataset and assigned 9000 samples for training and a 1000 for evaluation, see (**Figure 38**).

```
# Cell 4: Preprocess dataset - format prompt if needed, filter empty prompts

def format_example(example):
    # Example has keys: 'prompt', 'chosen', 'rejected' already
    # Optionally clean prompts here, but assume already clean

    # Filter empty or whitespace-only prompts
    if example["prompt"].strip() == "":
        return None
    return example

# Filter dataset with non-empty prompts
dataset = dataset.filter(lambda x: x["prompt"].strip() != "")
print(f"Total samples after filtering: {len(dataset)}")
```

```
Filter: 100%  10884/10884 [00:00<00:00, 80249.80 examples/s]
Total samples after filtering: 10884
```

```
# Cell 5: Explicit train/eval split with 9k train and 1k eval

# Shuffle dataset first (if not already shuffled)
dataset = dataset.shuffle(seed=42)

# Select first 10,000 samples only (if dataset is larger)
dataset = dataset.select(range(10000))

# Split manually
train_dataset = dataset.select(range(9000))
eval_dataset = dataset.select(range(9000, 10000))

print(f"Training samples: {len(train_dataset)}")   # Should print 9000
print(f"Evaluation samples: {len(eval_dataset)}")  # Should print 1000
```

```
Training samples: 9000
Evaluation samples: 1000
```

Figure 38-preprocessing for DPO

Process:

Lora config: After loading the dataset I defined the Lora configuration as such, again applying Lora is crucial here as that caused me many issues in this stage, you may encounter a trainable params zero error down the line, I'll guide you on how I fixed it, one major step was ensuring Lora is applied as in (**Figure 39**)**.** Another major point is to make sure you only have layers that were used in Lora in your SFT model, as that also caused issues i.e. if your sft model's Lora config did not have the target module up_proj and you use that in DPO, it could cause issues, secondly trainable params are dictated by the target modules.

```
import torch
from transformers import AutoModelForCausalLM, AutoTokenizer
from peft import PeftModel, PeftConfig, LoraConfig, get_peft_model, TaskType
from datasets import load_dataset
from trl import DPOTrainer, DPOConfig
import psutil
import numpy as np
import evaluate

lora_config = LoraConfig(
    r=4,
    lora_alpha=32,
    lora_dropout=0.05,
    target_modules=["q_proj", "v_proj", "k_proj", "o_proj", "gate_proj", "up_proj", "down_proj"],
    task_type=TaskType.CAUSAL_LM
)

base_model = get_peft_model(base_model, lora_config)
base_model.print_trainable_parameters()
# You can experiment with 'r', 'target_modules', dropout, etc.
```

```
trainable params: 3,153,920 || all params: 1,103,202,304 || trainable%: 0.2859
```

Figure 39-Lora config for DPO

DPO Config: We then define the DPO config accordingly as in (**Figure 40**), the parameters I messed around with were, per device train and eval batch, epochs, max steps, learning rate, however what I noticed was these changes didn't make most of the difference, the Lora parameters did, so focus on those if you want varying results.

```
dpo_config = DPOConfig(
    output_dir="./dpo_output10",
    beta=0.1,
    per_device_train_batch_size=3,
    per_device_eval_batch_size=2,
    gradient_accumulation_steps=4,
    num_train_epochs=2,
    max_steps=500,
    logging_steps=50,
    save_steps=100,
    eval_strategy="steps",
    eval_steps=100,
    learning_rate=1e-6,
    warmup_steps=10,
    fp16=True,
    max_prompt_length=512,
    max_length=768,
    seed=42,
    report_to="none",
)
```

Figure 40-DPO training args

Training function: We then run the training function as such, it's important to know that you've used the lora config in the function to ensure you applied it, furthermore the training gives this wide range of stats (**Figure 41**) which I will explain accordingly.

Steps basically dictate how many steps until an interval where the values are calculated for loss etc as shown, they get limited by the max steps value in training args. Training loss and validation loss are self explanatory, the lower the better. Rewards chosen means High reward for preferred responses so the greater this value gets the better. Rewards rejected means strong penalty for rejected responses (desirable in DPO), so the lower the better. Rewards accuracy means the odds of the model getting the correct response per step, closer to 1, the better. Rewards margin means the gap distinguishing between good and bad responses, greater the gap better the distinguishing ability. Logps chosen refers to the fluency/confidence of chosen outputs, higher the better, Logps rejected is the opposite so lower is better. The remaining two aren't that major. With these terms understood we can now analyze our model and truly see if its improving with the step intervals or not.

```python
trainer = DPOTrainer(
    model=base_model,
    ref_model=None,
    args=dpo_config,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
    processing_class=tokenizer,
    peft_config=lora_config,
)

trainer.train()
```

```
Extracting prompt in train dataset: 100% 9000/9000 [00:01<00:00, 6843.59 examples/s]
Applying chat template to train dataset: 100% 9000/9000 [00:00<00:00, 10844.94 examples/s]
Tokenizing train dataset: 100% 9000/9000 [00:17<00:00, 545.25 examples/s]
Extracting prompt in eval dataset: 100% 1000/1000 [00:00<00:00, 6453.33 examples/s]
Applying chat template to eval dataset: 100% 1000/1000 [00:00<00:00, 9980.00 examples/s]
Tokenizing eval dataset: 100% 1000/1000 [00:01<00:00, 556.07 examples/s]
No label_names provided for model class `PeftModelForCausalLM`. Since `PeftModel` hides base models input arguments, if label_names is not given, label_names can't be set automatically within `Trainer`. Note that empty label_names list will be used instead.
```

[500/500 1:03:26, Epoch 0/1]

Step	Training Loss	Validation Loss	Rewards/chosen	Rewards/rejected	Rewards/accuracies	Rewards/margins	Logps/chosen	Logps/rejected	Logits/chosen	Logits/rejected
100	0.131400	0.057282	1.072601	-2.177135	1.000000	3.249735	-397.145752	-399.569824	-3.405041	-3.455870
200	0.011600	0.008754	1.420105	-4.604169	1.000000	6.024273	-393.670746	-423.840149	-3.417531	-3.465865
300	0.003400	0.004756	1.478993	-5.600991	1.000000	7.079984	-393.081879	-433.808380	-3.420973	-3.469552
400	0.003800	0.003581	1.492437	-6.089572	1.000000	7.582010	-392.947388	-438.694153	-3.423343	-3.472959
500	0.002200	0.003280	1.496261	-6.257136	1.000000	7.753397	-392.909149	-440.369843	-3.423986	-3.474128

```
TrainOutput(global_step=500, training_loss=0.06890144874155521, metrics={'train_runtime': 3809.87, 'train_samples_per_second': 1.05, 'train_steps_per_second': 0.131, 'total_flos': 0.0, 'train_loss': 0.06890144874155521, 'epoch': 0.4444444444444444})
```

Figure 41-DPO trainer execution

Loading the adapter onto the base model: From what I understand, DPO acts as a layer over the base model instead of being an entire model itself, therefore it's necessary to apply this layer again, this exactly is what caused me hours of lost time as I wasn't sure of this step, I performed the following (**Figure 42)** yet still got zero trainable params as shown. The correct way is shown in (**Figure 43**), make sure the adapter name is distinct as that can cause issues.

```python
from peft import PeftModel

fine_tuned_model = PeftModel.from_pretrained(
    base_model,
    "/kaggle/working/dpo_output6/checkpoint-500",
    torch_dtype=torch.bfloat16
)
fine_tuned_model.train()
# List all modules with trainable params
for name, param in fine_tuned_model.named_parameters():
    if param.requires_grad:
        print(f"Trainable: {name}")

print(f"Trainable params: {sum(p.numel() for p in fine_tuned_model.parameters() if p.requires_grad)}")
```

```
Trainable params: 0
```

Figure 42-DPO Load attempt

```python
fine_tuned_model = get_peft_model(base_model, lora_config)
fine_tuned_model.load_adapter("/kaggle/working/dpo_output6/checkpoint-500", adapter_name="def3")
fine_tuned_model.set_adapter("def3")
fine_tuned_model.print_trainable_parameters()
```

```
trainable params: 1,531,904 || all params: 1,106,491,392 || trainable%: 0.1384
```

Figure 43-DPO Loading proper

Bleu function: This is semantically the same as that for SFT, only difference being I defined both models before the function.

Note: Only run the bleu function after (**Figure 43)** is done, otherwise you won't be using the DPO adapter you've prepared.

Bleu score: same as before, pass the same pairs into the function and get your relevant Bleu scores (**Figure 44**).

```python
print("Base Avg BLEU:", avg_base)
print("Fine-tuned Avg BLEU:", avg_fine_tuned)
```

```
Base Avg BLEU: 0.028412412520155406
Fine-tuned Avg BLEU: 0.034182949029769344
```

+ Code + Markdown

Figure 43-DPO Bleu score

Bert score: Same function as that for SFT (**Figure 44**)

```python
print(f"Average Base BERTScore: {avg_base_bert:.4f}")
print(f"Average Fine-Tuned BERTScore: {avg_fine_bert:.4f}")
```

```
Some weights of RobertaModel were not initialized from the model checkpoint at robe
You should probably TRAIN this model on a down-stream task to be able to use it for
Some weights of RobertaModel were not initialized from the model checkpoint at robe
You should probably TRAIN this model on a down-stream task to be able to use it for
Average Base BERTScore: 0.8512
Average Fine-Tuned BERTScore: 0.8503
```

Figure 44-DPO Bert score

Prompt response function: This function has a few semantic changes, namely the set adapter part, this is crucial to make sure your DPO model has the adapter set, secondly we can see the name def3 used, assume you have 2 different DPO adapters, if you set the name to default for both the system will confuse the two and choose one at random. To avoid this make sure to use distinct names when defining and to then use those exact names when accessing as in (Figure 45).

```python
def collect_model_responses_for_human_eval(pairs, base_model, fine_tuned_model, tokenizer, device):
    results = []

    # Set models to evaluation mode
    base_model.eval()
    fine_tuned_model.eval()

    # Activate adapter if available
    if hasattr(fine_tuned_model, "set_adapter"):
        fine_tuned_model.set_adapter("def3")

    for i, (prompt, expected_response) in enumerate(pairs):
        # Tokenize without truncating, no padding (to preserve prompt shape)
        input_ids = tokenizer(prompt, return_tensors="pt").input_ids.to(device)

        with torch.no_grad():
            # Generate from base model
            base_output = base_model.generate(
                input_ids=input_ids,
                max_new_tokens=150,
                temperature=0.7,
                do_sample=True,
                pad_token_id=tokenizer.pad_token_id,
                eos_token_id=tokenizer.eos_token_id
            )
            base_response = tokenizer.decode(base_output[0], skip_special_tokens=True)

            # Generate from fine-tuned model (DPO)
            fine_output = fine_tuned_model.generate(
                input_ids=input_ids,
                max_new_tokens=150,
                temperature=0.7,
                do_sample=True,
                pad_token_id=tokenizer.pad_token_id,
                eos_token_id=tokenizer.eos_token_id
            )
            fine_response = tokenizer.decode(fine_output[0], skip_special_tokens=True)

        results.append({
            "index": i,
            "prompt": prompt,
            "expected_response": expected_response,
            "base_response": base_response,
            "fine_tuned_response": fine_response
        })

    return results
```

Figure 45- DPO prompt response function

Saving DPO and adapter: To save the DPO model and adapter do the following as shown in {**Figure 46**} and (**Figure 47**). The checkpoint can be any point in the steps you prefer, i.e. if you feel the scores are plateauing after checkpoint 500, you don't need to load checkpoint 800 and so on.

```python
import shutil

shutil.make_archive("/kaggle/working/dpo_output6/checkpoint-500", 'zip', "/kaggle/working/dpo_output6/checkpoint-500")
```

```
'/kaggle/working/dpo_output6/checkpoint-500.zip'
```

Figure 46-DPO checkpoint zip

```python
adapter_save_path = "/kaggle/working/fine_tuned_adapter5"
fine_tuned_model.save_pretrained(adapter_save_path)

import shutil
shutil.make_archive("fine_tuned_adapter5", 'zip', root_dir=adapter_save_path)
```

```
'/kaggle/working/fine_tuned_adapter5.zip'
```

Figure 47-DPO adapter zip

Experimentation:

Lora config: Initially I had several errors caused because I forgot the step in (**Figure 43**), another misconception I read online when researching to fix this was that the target modules had to be the exact same for what you used in training as well. This is a major blunder and not at all a problem. From my own testing I figured that the major difference was brought upon due to the Lora configuration more than anything else.

DPO training args: Although I did test several combinations of these as well, they weren't nearly as fruitful in determining high value results and variations between models. The following three tables demonstrate the Lora config (**Table 1**), DPO training args (**Table 2**) as well as my 5 results (**Table 3**) for my 5 DPO models. Bleu scores were calculated at the time of prompt evaluation to ensure proper scoring i.e. (**Figure 48, Figure 49, Figure 50**). Some images attached for reference (**Figure 52**, **Figure 53**, **Figure 54**, **Figure 55**)

For reference, Base model SFT7's Bleu score is 0.0386 (**Figure 48**).

Table 1-Lora config- DPO

Model_ label	Rank	lora_alpha	Target _modules	lora_dropout	Trainable _params	Execution time
DPO1	8	32	"q_proj", "v_proj", "k_proj", “o_proj”, “gate_proj”, “up_proj”, “down_proj”	0.05	6.3 million	1:03:26
DPO2	8	16	"q_proj", "v_proj", "k_proj"	0.05	1.53 million	53:18
DPO3	8	16	"q_proj", "v_proj", "k_proj"	0.05	1.53 million	2:00:32
DPO4	16	32	"q_proj", "v_proj",	0.05	2.25 million	1:41:03
DPO5	8	16	"q_proj", "v_proj", "k_proj"	0.05	1.53 million	58:10

Table 2-Training args-scores-DPO

Model _ label	Training_batch _split_size	Epochs	Learning _rate	per_device_ train_batch	per_device_ eval_batch	Bleu score	Bert score
DPO1	9000/1000	3	5e-6	2	2	0.0353	0.8440
DPO2	9000/1000	3	5e-6	2	2	0.0372	0.8510
DPO3	9000/1000	3	3e-6	3	2	0.0338	0.8514
DPO4	4500/500	3	3e-6	3	2	0.0393	0.8525
DPO5	9000/1000	3	4e-6	2	2	0.0328	0.8503

Table 3-Training scores-DPO

Model_ label	Steps	Training _loss	Validation _loss	Rewards _chosen	Rewards_ rejected	Rewards_ Accuracies	Rewards_ Margins	Logps_ chosen	Logps_ rejected
DPO1	500	0.002	0.003	1.49	-6.25	1.00	7.75	-392.90	-440.36
DPO2	500	0.146	0.151	0.52	-1.45	1.00	1.98	-405.53	-397.23
DPO3	800	0.122	0.129	0.57	-1.60	1.00	2.17	-405.09	-398.71
DPO4	800	0.037	0.039	0.79	-3.06	1.00	3.86	-408.16	-416.72

DPO5	500	0.134	0.138	0.55	-1.56	1.00	2.05	-405.33	-397.67

```
print("SFT7 Avg BLEU:", avg_base)
print("DPO1 Avg BLEU:", avg_fine_tuned)

SFT7 Avg BLEU: 0.03863635743527098
DPO1 Avg BLEU: 0.0353183950787737
```

Figure 48-DPO1 and SFT7-Bleu comparison

```
results, avg_base, avg_fine_tuned = evaluate_model_bleu(pairs)

print("SFT7 Avg BLEU:", avg_base)
print("DPO1 Avg BLEU:", avg_fine_tuned)

SFT7 Avg BLEU: 0.0372758074776078
DPO1 Avg BLEU: 0.03381625469309938
```

Figure 49-DPO2 AND DPO3 Bleu scores (SFT7 is actually DPO2 and DPO1 is DPO3, forgot to change labels)

```
print("DPO4 Avg BLEU:", avg_base)
print("DPO5 Avg BLEU:", avg_fine_tuned)

DPO4 Avg BLEU: 0.0393402768662605585
DPO5 Avg BLEU: 0.032849301728591046
+ Code    + Markdown
```

Figure 50-DPO4 and DPO5 Bleu scores

```
--- Example 1 ---
Prompt:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I feel like no one really understands what I'm going through.
<|assistant|>


Expected Response:
I'm really sorry you're feeling that way. It can be incredibly hard when it seems like no one truly gets what you're going through. You're not alone, and your feelings are valid. I'm here to listen and support you.

DPO2 Response:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I feel like no one really understands what I'm going through.
<|assistant|>
I see your frustration. It's natural to feel like no one understands what you're going through. However, I'd like to offer a possible response to your sentence. Empathy is a strong and powerful emotion that can help you connect with others and find more understanding i
n situations. Here's an example response:

"I understand that it can be frustrating when people don't fully understand or empathize with what you're going through. It's normal to feel like you're alone and that no one else understands what you're going through. However, remember that it's okay to seek help and
support from friends or family members who can understand what you'

DPO3 Response:
<|system|>
How would you respond to this empthatically,Respond without using emoticons.
<|user|>
I feel like no one really understands what I'm going through.
<|assistant|>
Respond without emoticons:
I completely understand what you're going through. It's hard to understand what someone else is going through sometimes. However, I know that there are a lot of people out there who care and want to help.

I think it's important to stay in touch with others who can offer support. You might want to reach out to friends or family, or consider joining a support group or community center.

It's also important to remember that everyone has their own unique experiences and perspectives. Everyone's journey is different, and that's okay. And it's not uncommon for people to face struggles and challenges in life, no matter how long

------------------------------------------------------------
```

Figure 51-DPO prompt issue

```python
import torch
from transformers import AutoModelForCausalLM, AutoTokenizer
from peft import PeftModel, PeftConfig, LoraConfig, get_peft_model, TaskType
from datasets import load_dataset
from trl import DPOTrainer, DPOConfig
import psutil
import numpy as np
import evaluate

lora_config = LoraConfig(
    r=8,
    lora_alpha=32,
    lora_dropout=0.05,
    target_modules=["q_proj", "v_proj", "k_proj", "o_proj", "gate_proj", "up_proj", "down_proj"],
    task_type=TaskType.CAUSAL_LM
)

base_model = get_peft_model(base_model, lora_config)
base_model.print_trainable_parameters()
# You can experiment with 'r', 'target_modules', dropout, etc.
```

```
trainable params: 6,307,840 || all params: 1,106,356,224 || trainable%: 0.5701
```

Figure 52-DPO1 Lora config

```
dpo_config = DPOConfig(
    output_dir="./dpo_output5",
    beta=0.1,
    per_device_train_batch_size=2,
    per_device_eval_batch_size=2,
    gradient_accumulation_steps=4,
    num_train_epochs=3,
    max_steps=500,
    logging_steps=50,
    save_steps=100,
    eval_strategy="steps",
    eval_steps=100,
    learning_rate=5e-6,
    warmup_steps=10,
    fp16=True,
    max_prompt_length=512,
    max_length=768,
    seed=42,
    report_to="none",
)
```

Figure 53-DPO1- training args

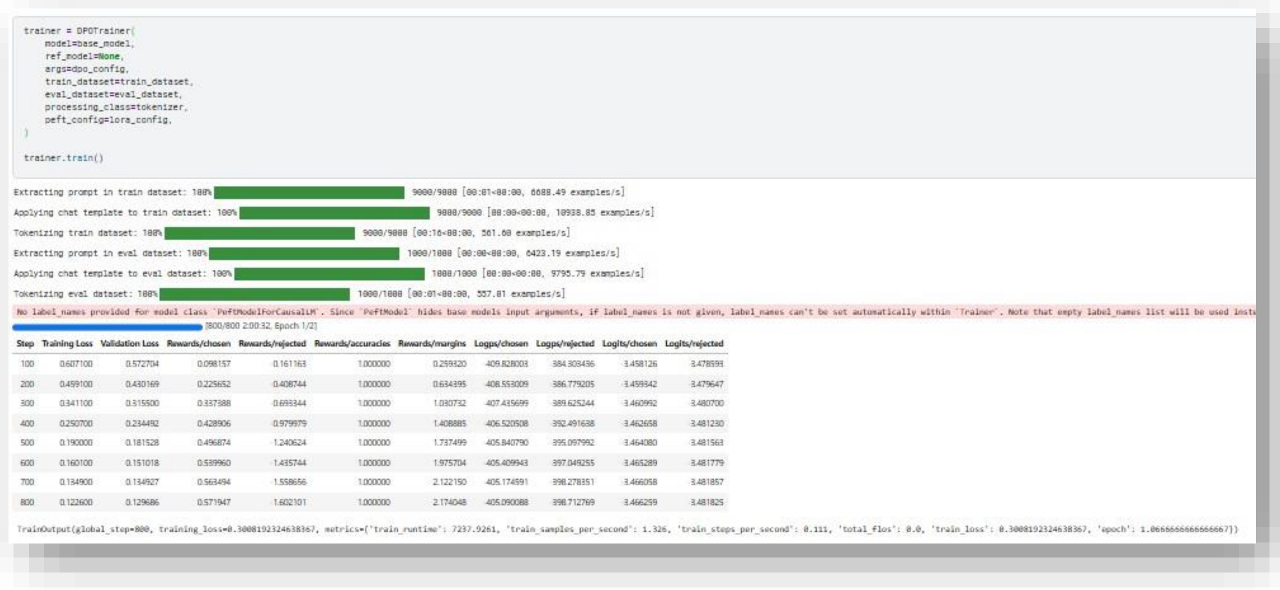
```
trainer = DPOTrainer(
    model=base_model,
    ref_model=None,
    args=dpo_config,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
    processing_class=tokenizer,
    peft_config=lora_config,
)

trainer.train()
```

```
Extracting prompt in train dataset: 100% 9000/9000 [00:01<00:00, 6843.59 examples/s]
Applying chat template to train dataset: 100% 9000/9000 [00:00<00:00, 10844.94 examples/s]
Tokenizing train dataset: 100% 9000/9000 [00:17<00:00, 545.25 examples/s]
Extracting prompt in eval dataset: 100% 1000/1000 [00:00<00:00, 6453.33 examples/s]
Applying chat template to eval dataset: 100% 1000/1000 [00:00<00:00, 9980.00 examples/s]
Tokenizing eval dataset: 100% 1000/1000 [00:01<00:00, 556.07 examples/s]
No label_names provided for model class `PeftModelForCausalLM`. Since `PeftModel` hides base models input arguments, if label_names is not given, label_names can't be set automatically within `Trainer`. Note that empty label_names list will be used instead.
[500/500 1:03:26, Epoch 0/1]
```

Step	Training Loss	Validation Loss	Rewards/chosen	Rewards/rejected	Rewards/accuracies	Rewards/margins	Logps/chosen	Logps/rejected	Logits/chosen	Logits/rejected
100	0.131400	0.057282	1.072601	-2.177135	1.000000	3.249735	-397.145752	-399.569824	-3.405041	-3.455870
200	0.011600	0.008754	1.420105	-4.604169	1.000000	6.024273	-393.670746	-423.840149	-3.417531	-3.465865
300	0.003400	0.004756	1.478993	-5.600991	1.000000	7.079984	-393.081879	-433.808380	-3.420973	-3.469552
400	0.003800	0.003581	1.492437	-6.089572	1.000000	7.582010	-392.947388	-438.694153	-3.423343	-3.472959
500	0.002200	0.003280	1.496261	-6.257136	1.000000	7.753397	-392.909149	-440.369843	-3.423986	-3.474128

```
TrainOutput(global_step=500, training_loss=0.06890144874155521, metrics={'train_runtime': 3809.87, 'train_samples_per_second': 1.05, 'train_steps_per_second': 0.131, 'total_flos': 0.0, 'train_loss': 0.06890144874155521, 'epoch': 0.4444444444444444})
```

Figure 54-DPO1-training execution


```
trainer = DPOTrainer(
    model=base_model,
    ref_model=None,
    args=dpo_config,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
    processing_class=tokenizer,
    peft_config=lora_config,
)

trainer.train()
```

```
Extracting prompt in train dataset: 100% 9000/9000 [00:01<00:00, 6688.49 examples/s]
Applying chat template to train dataset: 100% 9000/9000 [00:00<00:00, 10938.85 examples/s]
Tokenizing train dataset: 100% 9000/9000 [00:16<00:00, 561.60 examples/s]
Extracting prompt in eval dataset: 100% 1000/1000 [00:00<00:00, 6423.19 examples/s]
Applying chat template to eval dataset: 100% 1000/1000 [00:00<00:00, 9795.79 examples/s]
Tokenizing eval dataset: 100% 1000/1000 [00:01<00:00, 557.81 examples/s]
No label_names provided for model class `PeftModelForCausalLM`. Since `PeftModel` hides base models input arguments, if label_names is not given, label_names can't be set automatically within `Trainer`. Note that empty label_names list will be used inste
[800/800 2:00:32, Epoch 1/2]
```

Step	Training Loss	Validation Loss	Rewards/chosen	Rewards/rejected	Rewards/accuracies	Rewards/margins	Logps/chosen	Logps/rejected	Logits/chosen	Logits/rejected
100	0.607100	0.572704	0.098157	-0.161163	1.000000	0.259320	-409.828003	-384.303436	-3.458126	-3.478593
200	0.459100	0.430169	0.225652	-0.408744	1.000000	0.634395	-408.553009	-386.779205	-3.459342	-3.479647
300	0.341100	0.315500	0.337388	-0.693344	1.000000	1.030732	-407.435699	-389.625244	-3.460992	-3.480700
400	0.250700	0.234492	0.428906	-0.979979	1.000000	1.408885	-406.520508	-392.491638	-3.462658	-3.481230
500	0.190000	0.181528	0.496874	-1.240624	1.000000	1.737499	-405.840790	-395.097992	-3.464080	-3.481563
600	0.160100	0.151018	0.539960	-1.435744	1.000000	1.975704	-405.409943	-397.049255	-3.465289	-3.481779
700	0.134900	0.134927	0.563494	-1.558656	1.000000	2.122150	-405.174591	-398.278351	-3.466058	-3.481857
800	0.122600	0.129686	0.571947	-1.602101	1.000000	2.174048	-405.090088	-398.712769	-3.466259	-3.481825

```
TrainOutput(global_step=800, training_loss=0.3008192324638367, metrics={'train_runtime': 7237.9261, 'train_samples_per_second': 1.326, 'train_steps_per_second': 0.111, 'total_flos': 0.0, 'train_loss': 0.3008192324638367, 'epoch': 1.0666666666666667})
```

Figure 55-DPO3-training execution

Prompt scoring criteria: Same as that in SFT. An issue I realized here was that my instructions while they worked for SFT, they caused issues with DPO, i.e. When instructed "How would you respond to this empathetically?" (**Figure 51**) The model instead took it as me asking how I would respond to someone else that said the prompt, therefore I corrected it to "Respond to this empathetically".

DPO1:

Prompt	Helpfulness	Harmlessness	Relevance	Response Quality
1	5	5	5	5
2	5	5	5	5
3	5	5	5	5
4	5	5	5	5
5	2	1	1	1
6	5	5	4	5
7	4	4	4	3
8	5	5	5	5
9	5	4	4	4
10	5	5	5	5

Averages:

Helpfulness: 4.6

Harmlessness: 4.4

Relevance: 4.3

Response Quality: 4.3

Total avg score: 4.4

DPO2:

Prompt	Helpfulness	Harmlessness	Relevance	Response Quality
1	5	5	5	5
2	4	4	3	3
3	5	5	5	5
4	5	4	4	4
5	2	3	3	1
6	5	5	4	4
7	5	5	5	4
8	4	3	4	3
9	4	3	5	3
10	5	5	5	5

Averages:

Helpfulness: 4.4

Harmlessness: 4.2

Relevance: 4.3

Response Quality: 3.7

Total avg score: 4.15

DPO3:

Prompt	Helpfulness	Harmlessness	Relevance	Response Quality
1	5	5	5	5
2	4	5	4	4

3	5	5	5	5
4	4	4	5	4
5	4	3	4	3
6	1	3	2	1
7	1	4	1	2
8	5	5	5	5
9	5	5	5	5
10	4	5	5	4

Averages:

Helpfulness: 3.8

Harmlessness: 4.5

Relevance: 4.1

Response Quality: 3.8

Total avg score: 4.05

DPO4:

Prompt	Helpfulness	Harmlessness	Relevance	Response Quality
1	3	4	3	3
2	3	3	4	4
3	5	5	5	5
4	4	5	3	3
5	1	3	3	2
6	5	5	5	5
7	5	5	4	4
8	5	4	5	5
9	5	5	4	5
10	5	5	5	5

Averages:

Helpfulness: 4.1

Harmlessness: 4.4

Relevance: 4.1

Response Quality: 4.1

Total avg score: 4.175

DPO5:

Prompt	Helpfulness	Harmlessness	Relevance	Response Quality
1	4	3	5	4
2	4	5	4	3
3	5	5	5	4
4	5	5	5	5
5	1	2	1	1
6	5	4	5	5
7	5	5	5	5
8	5	4	4	4
9	3	4	3	3
10	5	4	5	5

Averages:

Helpfulness: 4.2

Harmlessness: 4.1

Relevance: 4.2

Response Quality: 3.9

Total avg score: 4.1

Assessment:

Lora config: Changing rank didn't affect the scores much, although DPO4 saw a significant boost and has a higher rank i.e. 16, it's actually the target modules which make the real difference, these modules when increased usually meant more trainable parameters for the model. Another interesting fact was that the best training results were from DPO1, which had the exact same Lora configuration as the base trained model.

DPO4 was a strange model, while having the highest outright bert and bleu score, higher than even the base sft model yet it was trained on half of the data the others were trained on. However, scores alone aren't the be all and end all as upon examining the prompt responses we can clearly see DPO1 which had the lowest comparative bert score outdid the rest by a mile in terms of prompt responses.

Training args: Here epochs decided the training time and also how much we aimed to allow our loss values to decrease, epochs could definitely have gone up to 1000 or 1500 but lack of resources limited that side of experimentation. Learning rate's effects were different where a lower learning rate at times gave a higher score and at times a lower score, rendering it not so effective in comparison to other factors from what I saw. Per device train batch also didn't have much major effect though I expected it to as it allows more training samples per execution.

Prompt score analysis:

All the models had at the very least an avg score of 4 which is a big improvement over our SFT models however as the base model was that strong this was to be expected. Prompt 5 for some reason made every single model struggle, be it my SFTs or my DPO models, which I'm still not too clear on. Overall, I feel that for me personally my SFT7 was the best model in general. While not having the highest Bleu score out of all 10 models it performed the best in prompt responses. DPO1 was a close second but since it had SFT7 as a base and ended up underperforming still wasn't a good sign. While DPO enables the prompts to feel more conversational like you're talking to a human on the other end, it also brought upon confusion where at times it started treating it as a convo and even spoke for the user which was never asked for, secondly, it outright mentioned empathy and the definitions of empathy in some models, I didn't face either of these issues with any of my SFT models.

The rest of the DPO models all faced the same issue where they were really good at answering some prompts and then really bad with others, comparatively DPO1 was the most consistent throughout.

Best overall model: SFT7

Higher bleu and bert scores than my base tinyllama model, best responses generated across all models, gained more humanity in its responses as opposed to the base model, as well as empathetic speech patterns and understanding of how to respond to make sure you make the user feel seen and heard, offer advice on how to cope with the issue or move on, and finally offer words of encouragement like good luck!

Comparison of SFT and DPO:

- In prompt responses DPO was better but that's because our base model was already so well formed
- The training results of DPO were much more in depth as opposed to those defined by SFT.
- I preferred the application of DPO, i.e. instead of being a full model it's a layer that goes on top of an existing model thus reducing memory requirements.
- Preferred the stepwise visualization in DPO over the epoch-based visualization of SFT.
- DPO removed some of the features fine tuning added i.e. prompt specific wording, this could be seen in the bleu scores as well where the performance deteriorated as the model became a bit too generalized.
- SFT was overall better for my empathetic model approach however DPO talked more like a human which could be a preference for some.
- Training args had much more importance in SFT than DPO which favored Lora configuration heavily.
- DPO was also much more complicated to implement as I couldn't find much helpful information to fix the errors I faced as in my experience SFT is more well documented.

Dataset citations:

Humanlike dialogues:

Çalık, Ethem Yağız, and Akkuş, Talha Rüzgar. “Enhancing Human-like Responses in Large Language Models.” *ArXiv.org*, 2025, arxiv.org/abs/2501.05032. Accessed 10 June 2025.

Empathetic dialogues:


```
@inproceedings{rashkin-etal-2019-towards,
   title = "Towards Empathetic Open-domain Conversation Models: A New Benchmark and Dataset",
   author = "Rashkin, Hannah  and
     Smith, Eric Michael  and
     Li, Margaret  and
     Boureau, Y-Lan",
   booktitle = "Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics",
   month = jul,
   year = "2019",
   address = "Florence, Italy",
   publisher = "Association for Computational Linguistics",
   url = "https://aclanthology.org/P19-1534",
   doi = "10.18653/v1/P19-1534",
   pages = "5370--5381",
}
```