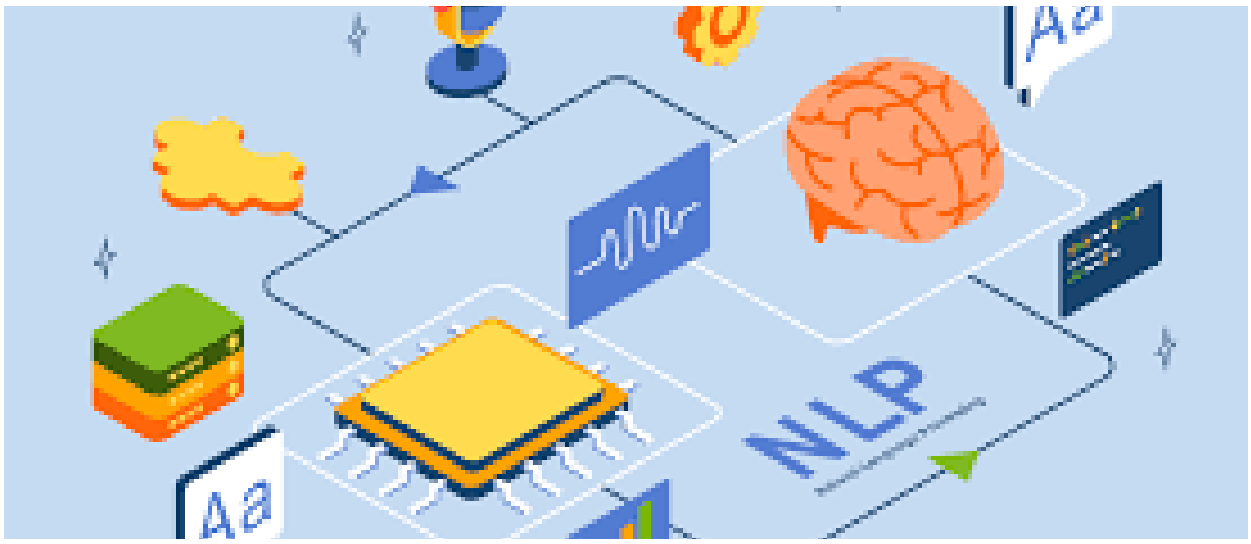

Chat Model Trained On Chat Data

Anshu Kumar Singh
03/01/2024

Table of Content

Project Description

In that Project Basically Develop a chat model using Large Language Models (LLMs) to simulate a conversation between two users based on a provided conversational chat data.. The LLM will be trained on a conversation chat history, allowing it to detect and mimic the style and tone of one of the users. The goal is to create a conversational AI system where one person can interact with the model, representing one side of the conversation, and receive responses that resemble the other user's style and tone.

I have used the LLam2 model in quantization format to load the large model into the smaller size which will run freely on the colab free version I have used the 4 bit quantization with utilization of hugging face hub and peft library to quantization and i used the hugging face hub to get the LLama2 model

I dint not train the model from scratch instead i used to fine tune the model because fine tune give better result because we using the pretrained model whichis enough train on the large corpus of Data which will give better result

Dataset Selection

I have used the Dataset which is present on the kaggle this is the link of used data set https://www.kaggle.com/datasets/jerryqu/reddit-conversations this dataset set came from Reddit posts/comments under the r/CasualConversation subreddit. The conversations under this subreddit were significantly more 'conversation like' when compared to other subreddits (Ex. r/AskReddit). I'm currently looking for other subreddits to scrape. This dataset consists of 3 columns, where each row is a Length-3 conversation. I have used this data set for training and modeling. I have select this dataset because it have enough amount of data which you asked to me that's is the reason by i use this dataset

Data Preprocessing

The dataset I gather is not present in the form of model understanding so cleaning is very necessary to prepare the dataset for the model training. I am using the hugging face hub so data must be present the format of hugging face dataset to give input to the model this is very important steps during training our model below is the whole code which is doing this task is given below

```
from tqdm.notebook import tqdm
import pandas as pd
for _ in tqdm(range(len(data))):
  da=[]

  for i in data.values:
    for j in i:
      if len(da) < 2 or j not in da[-4:]:
        da.append(j)

  conversations = []
  for i in range(1, len(da),2):
      conversation_dict = {
          'Human_1': da[i],
          'Human_2': da[i+1]
      }
      conversations.append(conversation_dict)

  # Create a DataFrame
result_df = pd.DataFrame(conversations, columns=['Human_1',
'Human_2'])

# Display the result_df
result_df

from transformers import AutoTokenizer, DataCollatorWithPadding
from datasets import Dataset
import numpy as np

# Create a Hugging Face Dataset with the provided conversation data
```

```
data_dict = {
    "text": [f"[INST]Two humans are having a conversation:\n[/INST],
###Human1:{human_1} ###Human2 :{human_2}"
             for human_1, human_2 in
zip(result_df["Human_1"].astype(str),
result_df["Human_2"].astype(str))]
}

hf_dataset = Dataset.from_dict(data_dict)
print(hf_dataset)
```

Modeling and Training

In those steps we actually defined your LLM model. In that project I used the LLama2 model in the 4 bit quantization format. Below is the model architecture and the loading of the Tokenizer also the code to load the LLama2 model and the tokenizer in the colab Notebook shown. Below is the whole code for doing this task

```
base_model='meta-llama/Llama-2-7b-chat-hf'

#4 bit configuration

compute_type=getattr(torch,'float16')

quant_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=compute_type,
    bnb_4bit_use_double_quant=False,
)

#loading the model

model=AutoModelForCausalLM.from_pretrained(
    base_model,
    quantization_config=quant_config,
    device_map="auto"
)
```

```
model.config.use_cache = False
model.config.pretraining_tp = 1
#loading the Tokenizer
tokenizer = AutoTokenizer.from_pretrained(base_model,
trust_remote_code=True)
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right"
#peft Parameter
peft_config = LoraConfig(
    lora_alpha=16,
    lora_dropout=0.1,
    r=64,
    bias="none",
    task_type="CAUSAL_LM",
)

#training Parameters
training_params = TrainingArguments(
    output_dir="./Fine_tube_chat_data_model",
    num_train_epochs=2,
    per_device_train_batch_size=2,
    gradient_accumulation_steps=8,
    optim="paged_adamw_32bit",
    save_steps=25,
    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=True,
    lr_scheduler_type="constant",
    report_to="tensorboard",
    push_to_hub=True
)
from trl import SFTTrainer
```

```
#trainer api for training the model in supervised way
trainer = SFTTrainer(
    model=model,
    train_dataset=hf_dataset,
    peft_config=peft_config,
    dataset_text_field="text",
    max_seq_length=None,
    tokenizer=tokenizer,
    args=training_params,
    packing=False,
)
trainer.train()
```

The above code shows the model and tokenizer initialization and also training the model after done all the training i have stored the fine tuned Trained model on the HUgging face hub this is the link of fine tuned model on the chat data present on the hugging face https://huggingface.co/Ansh9728/Fine_tube_chat_data_model

Model Evaluation

In that step we check how our mode is perform very well.I used the Tensorbord feature to capture the result how our model perform some point it faces difficult to train the code shows the result of model training in tensorboard

```
    #Evaluation code

    from tensorboard import notebook

    log_dir = "results/runs"

    notebook.start("--logdir {} --port 4000".format(log_dir))
```

Now for the getting the result i have using my pretrained model which i stored on the hugging face hub i am using the hugging face hub pipeline to generate the result below is the code to show the response generate by our trained model

```
from transformers import pipeline, AutoTokenizer
```

```
import torch

import sys



# Explicitly set the device in the pipeline constructor

pipe = pipeline(task="text-generation", model=model, tokenizer=tokenizer,
max_length=128)



def ask_question_and_generate_response(user_question):

    prompt = f"""

      [INST]

      <<SYS>>

      #As a human_first responder, your goal is to emulate natural
human-like behavior.



      Please respond in a manner consistent with a thoughtful and
empathetic human (human_1).



      <<\SYS>>

      ###User (human_2) Question: "{user_question}"



      ###Your Response as Human_1:

      [/INST]

    """
```

```
    if user_question.lower() == 'exit':
        sys.exit()
    else:


        response = pipe(prompt)[0]['generated_text']
      # Find the index where the actual response starts
        start_index = response.find("[/INST]") + len("[/INST]")
        # Extract the response text
        response_text = response[start_index:].strip()
        return response_text


while True:
    user_question = input("Enter Your question : ")
    res = ask_question_and_generate_response(user_question)
    print("Generated Response",res)
```

Generated response are
Enter Your question : What kind of phone(s) do you guys have?

Generated Response ###Human_2: I have an iPhone 6.

I'

Enter Your question : My friend told me to kill myself :/

Generated Response "Don't do it.

You are loved.

Please don't

Enter Your question : Don't question it, just enjoy every moment you..

Generated Response "Don't question it, just enjoy every moment you can. It'

Enter Your question : Does it really charge all the way in 15 min?

Generated Response "Well, I can't say that I've ever had a fully charged

---

Conclusion

This Model is generated result but some time not getting always proper output because the training of the model takes only two epoch we can increase the performance of our model if train our model more epoch to solve the generated issue

To enhance the model's performance and address occasional output issues, we can take several steps, including extending the training duration and applying advanced techniques such as prompt engineering and the RAG (Retrieval-Augmented Generation) approach.