1. I started researching models for fine-tuning around the dataset provided. In the past, I had used the Llama 2 7B chat version to fine-tune a conversational bot focused on medical data, but it didn't perform well. This time, I was considering trying a simpler model.

2. I researched and found T5 and BART to be good options, with Llama 2 as a last resort if both fail, given the smaller dataset. These models are also easier to train using the free version of Colab. I posted the same question on a subreddit, asking for recommendations on my approach. The responses suggested trying the T5 model first, and if the results aren't satisfactory, switching to Llama 2. This aligned with my original plan and reinforced my decision.

3. I read several articles on fine-tuning a T5 model and began working on it. The HuggingFace forum provided useful insights on hyperparameters. I chose T5-small, as it performs well on Colab and also given our smaller dataset of around 7,182 rows, opting for T5-small seemed like a good decision.

4. I downloaded the dataset with the goal of creating a fine-tuned model to upload on HuggingFace, making it easier for the ChataAI team to run inferences. I began training the model using the provided dataset split, carefully choosing hyperparameters and various text generation metrics to ensure I was on the right track. I also created some random examples and tested the model on 5 samples from the test dataset. This helped me assess whether I was moving in the right direction.

5. The initial parameters I chose for the trainer were a batch size of 16, a learning rate of 1e-5, and 5 training epochs. This configuration resulted in a validation loss of 0.25 for the final epoch, with ROUGE metrics above 90, a BERTScore of 90, and a METEOR score around 91. These three metrics provided a comprehensive evaluation of the model's performance, each targeting a different aspect of the generated text.

6. I started hyperparameter fine-tuning and determined that a batch size of 8, a learning rate of 3e-4, and 3 training epochs produced excellent results. This configuration achieved a training loss of around 0.15 and a validation loss of 0.21, with a ROUGE score above 94, BERTScore of 93, METEOR score of 95.

7. Increasing the number of epochs beyond 5 further reduced the training loss, but I was concerned it might lead to overfitting on the training set and potentially cause overconfidence. To address this, I experimented with different variations to improve the score without affecting the overall model confidence, and found that 4 epochs was a good balance.