Fine-tuning an LLM for German-French Language Translation

Alireza Poostforoosh
Matriculation Number: 431005

May 8, 2025

1 Introduction

In this task we have to finetune a pretrained general purpose large language model with at least 1 Billion instructions to increase it's performance in translating form German to French. We have to find a proper German to French dataset for our LLM to finetune on and the appropriate model required for this task that will satisfy our task criteria.

1. The model must have at least 1 Billion instructions
2. The model should be a general purpose language model not a translation model
3. We should be able to fine tune this model on free version of Google Colab Environment

2 Research Phase

2.1 Dataset Selection

I have decide to use "German-French website parallel corpus from the Federal Foreign Office Berlin" Link to the dataset web page. This dataset contained 11,852 pairs that were translated between October 2013 and early November 2015. The reason behind choosing such dataset was the fact that this dataset was published by an official organization, therefor I could be certain that the translations must have a decent accuracy, however, due to the news-based nature of this dataset it might slightly suffer from lacking diversity in the topics covered in the dataset.

2.2 Model Selection

First I have started experimenting with GPT 2 XL with 1.68B parameters but due to lower instructions it wasn't able to achieve significant accuracy therefor I have decided to use the Microsoft Phi-2 model with 2.78B instructions.

In addition to having more instructions, based on hugging face website, Phi-2 is performing better on detecting languages Phi-2 showcased a nearly state-of-the-art performance among models with less than 13 billion parameters. By comparing GPT 2 XL which was only trained on English, I decided to proceed my fine tuning with Phi-2

3 Design Phase

3.1 Fine-tuning Approach

First due to limited computational resources available in free version of Google Colab and large size of the mentioned models, I had to use PEFT (Parameter-Efficient Fine-Tuning) with LoRA (Low-Rank Adaptation) to reduce computations and memory usage.

For instance, I have used r=32 (Rank) to allow LoRA layers to capture more complex features. Although this choice of Rank would increase my computational costs but it is still more efficient than fine-tuning on the entire model.

I have used lora_alpha=64 (Scaling Factor) to amplify adaptations and enhance the influence of my training and have a more effective fine tuning and also, by using lora_dropout=0.1 (Dropout) I wanted to ensure that my fine tuned model wouldn't overfit and to moderate regularization.

I have also decided to Quantize my model to 8 bits to ensure lower GPU memory consumption and be able to fine tune this model on limited respurces provided by Google Colab.

3.2 Data Splitting

I used 80/20 percent for my data splitting because this ratio has been tested and trialed before in many models and it is considered a safe approach.

3.3 Prompt Design

I have prompted the fine tuned model "Translate the following German text into French:" because I wanted to clearly instruct the model about the exact translation task. I believe this prompt would clearly instruct the model to produce a relatively accurate outputs without clutter.

3.4 Evaluation Metric

I have decided to use BLEU and BERTScore as my evaluation metrics, because BELU measures translation quality based on word overlap with reference translation and it it is a reliable standard for machine translation tasks. BERTScore would also complement BELU by evaluating similarities through contextual embeddings which would ensure that translations are not only lexically correct but also semantically coherent.

4 Implementation Phase

4.1 Data Loading and Splitting

I have used BeautifulSoup for tmx parsing and then extracted the translation units, then validate my (German-French) language pairs and then randomly sampled 1000 pairs from Dataset A and saved them into dataset_a.json and split it into my training and test sets(80% training and 20% test) and saved them into dataset_a_test.json and dataset_a_train.json.

4.2 Evaluation of Model A

My baseline performance before any fine tuning of Model A on my test set is described like this **BLEU score**: 10.290348648040435 and **Average BERTScore F1**: 0.6385698318481445

Based on these scores and observing the model's translations it was obvious that although model was able to produce some instances of slightly successful translations from German to French but it was struggling to complete this task. I have also devised a method by using langdetect to remove the part of the model's output which was not in French. This method was looped through for maximum of five times to try to encourage the model to produce another output therefor increasing the likely hood of a correct and all French translation free of any clutter. In this post processing method, I have also tried to remove the part of the output which is not in French to again, increase the likely hood of producing a less cluttered output.

In this case, even by using such post processing methods, the model struggled to produce a good output.

4.3 Fine-tuning and Evaluation of Model B

I have fine tuned model A with these LoRA parameters(r=32, lora_alpha=64, lora_dropout=0.1) which I have explained the reasoning behind these parameters above. However I encountered mixed evaluation results during my fine tuning even with the same parameters. The model was trained for 5 epochs.

I don't know the exact reason behind such behaviors but my BELUScore in my last fine tuning attempt is : **BLEU score**: 11.102294159296243 , **Average BERTScore F1**: 0.7070519328117371. These evaluation metrics have both increased with 5 from Model A. (BELUScore increased by about 0.81 indicating an enhanced lexical and Phrase level translation and Average BERTScore F1 increased by about 0.068 showing an improved semantic and contextual accuracy.)

During translation production stage I have also used the filtering post processing method of French sentences explained above to increase my model's performance.

4.4 Synthetic Data Generation

I have created Dataset B by using SambaNova API and used Meta Llama 3.3 with 70B instructions. Due to limitations on api calls I have encountered during producing my synthetic dataset B, I had to collect my dataset in 2 parts and append them to each other and then I searched for duplicates or corrupted files as well as if there is any missing translations in my synthetic dataset B.

I have also tired multiple prompts but after trial and error I decided to explicitly prompt the model with some instances of the exact type of translation pairs that I need and then I got better outputs from the model.

4.5 Fine-tuning and Evaluation of Model C

After fine tuning Model A on Dataset B, which was generated using Meta Llama 3.3 with 70B instructions, Model C was produced and with PEFT configurations as my previous models,(r=32, lora_alpha=64, lora_dropout=0.1).

After 3 epochs my model showed yet another improvement compared to previous models as I expected. On model C, **BLEUScore**= 11.279843877349599, which indicates another improvement (by roughly 0.18 from model B) in lexical accuracy of my model and translation fluency of the Model C compared to Model A and B.

Average BERTScore F1: 0.7110763788223267 has also increased slightly compared to Mode B (roughly 0.004 increased compared to Model B) showing a better semantic and contextual coherence.

4.6 Dataset Combination

Dataset C was created by combining Dataset A's training subset with synthetic Dataset B resulting a larger training dataset. The combined dataset was shuffeled with a random seed. This ensures a well distributed mixture of Dataset A and the synthetic dataset B.

4.7 Fine-tuning and Evaluation of Model D

I have trained Model C for 2 epochs with such LoRa Configurations, r=8 which was reduced from previously used 32 so this model could train on my limited computational resources, although this decision would provide fewer trainable parameters but I had to use it to combat my limited resources. I have also used lora_alpha=32 and lora_dropout=0.1 as explained before. After evaluating my model, I got **BLEUScore**:10.813005337959174 which was decreased slightly compared to Model C and Model B (roughly 0.47 lower than Model C and 0.29 lower than Model B) but was still higher than Model A (roughly 0.52 higher than Model A).**Average BERTScore F1**: 0.7110893726348877 which was almost identical to Model C and marginally improved from model B.

This indicates minor dilution of lexical precision was visible however, the consistent BERTScore indicates that semantic quality and contextual understanding remained stable confirming that this dataset helped preserve translation relevance.

5 Results Visualization

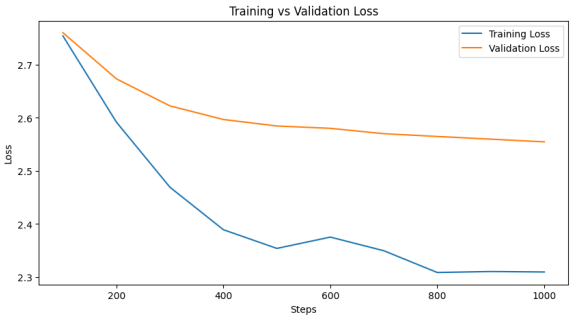

Figure 1: Model B Training vs Validation Loss

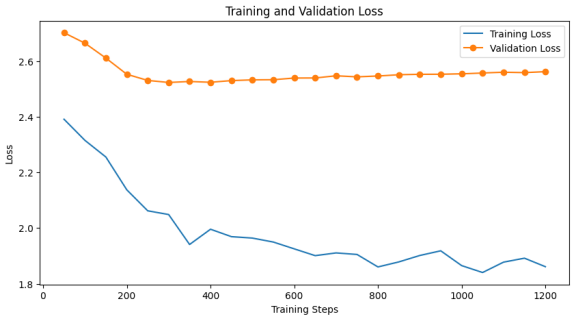

Figure 2: Model C Training vs Validation Loss

Figure 3: Model D Training vs Validation Loss

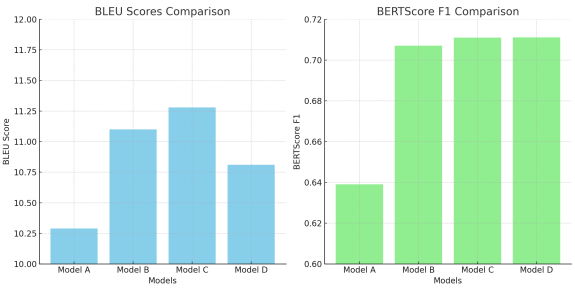

Figure 4: Comparison of Models' BLEU Scores and BERTScore F1