Finetuning Large Language Model for Nepali Health-Related Question Answering

Chhabilal Acharya[1], Manoj Kumar Guragai[2], Pukar Karki[3], and Yunisha Baskota[4]

[1] *Department of Electronics & Computer Engineering, Purwanchal Campus, IOE, Tribhuvan University, Dharan, Nepal*

April 17, 2024

Abstract: This paper explores the advancement of question-answering systems leveraging state- of-the-art natural language processing capabilities, specifically Bidirectional and Auto- Regressive Transformers (BART), Multilingual pre-trained Text-To-Text Transformers (T5), and Llama-2-7b. The study focuses on training and fine-tuning various pre- trained language models, including mT5-small, llama-2-7b with PEFT, and QLORA on diverse Nepali question-answering datasets. The evaluation metrics encompass BLEU scores, training and validation losses, and generation lengths. The performance of the mT5-small models showcase their potential. Initially, the model achieved a BLEU score of 11.9057, with validation and training losses of 1.7343 and 1.9988, respectively, and a generation length of 256 tokens after one epoch. Through continuous training over 100 epochs, improvements were observed, with the BLEU score reaching 80.3484 in the 125th epoch, alongside reduced validation and training losses (1.1593 and 1.3723, re- respectively), and an increased generation length of 512 tokens. The integration of Qlora and PEFT during Llama2 model training further enhanced its performance, particularly in responding to questions.

Keywords: Question-Answering Systems, Natural Language Processing, Transformers, mT5, BART, Llama2, BLEU Scores, ROUGE Scores

1 Introduction

In the field of Natural Language Processing (NLP), our project aims to create an open-source Nepali Health-QA Language Model using fine-tuned Transformers We seek to satisfy the growing demand for accurate and easily available health information in Nepali, acknowledging the critical role of natural language comprehension in allowing meaningful health-related conversations.

Our model not only answers questions, but it also provides vital second opinions and thorough insights into healthcare issues. The project consists of three major components: dataset curation, model building, and extensive analysis.

We diligently compile a dataset of health-related questions and answers in preparation for training and testing our question-answering algorithm. To refine the dataset and improve the model's performance, a variety of preprocessing techniques are used, including sampling, filtering, and tokenization.

The mT5 model is extensively fine-tuned, with a focus on health-related concerns. Iterative changes with mT5-small models assure the model's ability to generate reliable responses. Pretraining and finetuning techniques are used to improve the model's knowledge of the Nepali language's health-related nuances.

Quantitative analysis, like as BLEU and rouge scores, supplements qualitative human review to create a strong assessment framework. Human evaluators, including experts and users, evaluate response quality using criteria such as naturalness, relevancy, and overall quality. This integrated method allows for a full examination of the model's ability to generate precise and contextually relevant responses to health-related questions.

2 Related Works

In recent years, the landscape of Natural Language Processing (NLP) has witnessed significant advancements driven by various pioneering models and methodologies. In recent years, Natural Language Processing (NLP) has seen remarkable progress with the introduction of pioneering models and techniques. "Attention is All You Need" (2017)[11] introduces the transformer architecture, revolutionizing NLP by relying solely on attention mechanisms. This departure from traditional recurrence and convolution-based models enhances the model's ability to handle long-range dependencies, facilitating improved parallelization and bidirectional relationship modeling. "mt5: A massively multilingual pre-trained text-to-text transformer" (2020)[14] presents the massively multilingual text-to-text transformer (mT5). Tailored for multilingual applications, this model follows a text-to-text framework, contributing to the development of versatile language models capable of handling diverse linguistic contexts. NepBERT: This model, developed in 2021, is a masked language model (MLM) pre-trained on a large corpus of Nepali text. It demonstrates promise in various NLP tasks, including sentiment analysis and named entity recognition. NepBERTa: An improvement over NepBERT, released in (2022)[9], also trained on a large Nepali corpus. This model achieves state-of-the-art performance on several Nepali NLP benchmarks compared to previous models. "Nepali Encoder Transformers: An Analysis of Auto Encoding Transformer Language Models for Nepali Text Classification" by Manish Bhatta et al. (2022)[7] delves into the exploration of pre-training Transformer models for Nepali text classification tasks. "Llama 2: Open foundation and fine-tuned chat models" (2023)[10] takes a distinctive approach, focusing on an open foundation and fine-tuned chat models. The model, named Llama 2, is designed to offer versatility and customization through fine-tuning, making it an appealing choice for various chat-based applications. "Chatlaw: Open-source legal large language model" (2023)[2] introduces an open-source legal large language model with a primary focus on enhancing legal language understanding. The model architecture and specific features remain unspecified, highlighting its dedication to catering to the intricacies of legal discourse. "Qlora: Efficient finetuning of quantized LLMs" (2023)[3] explores the realm of efficient fine-tuning for quantized Large Language Models (LLMs). This approach addresses the challenges associated with quantized models, emphasizing enhanced efficiency in the fine-tuning process for improved model performance.

3 Method

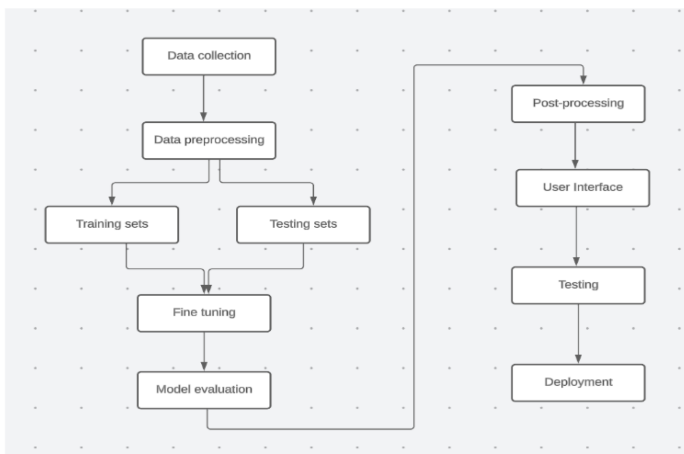

Figure 1: Methodology

Our project technique includes numerous crucial stages. Initially, we obtain relevant datasets via extensive data collection operations. Subsequently, we rigorously preprocess the collected data to ensure purity and consistency. The preprocessed data is then divided into separate training and testing sets for further analysis. The model fine-tuning procedure involves adapting a pre-trained model to our unique domain utilizing the training set. The fine-tuned model's performance against the testing set is assessed to determine its efficacy and correctness. Finally, post-processing methods are used to adjust the model's output and improve the quality of the generated text. The difficulties presented by Nepali language resources have encouraged novel solutions, such as the translation of health-related datasets from other languages.

This strategy, together with tokenization and model fine-tuning focusing on mT5 and Llama2 models, serves as the foundation of our methodology. We intend to optimize the model for generating trustworthy and accurate health-related text in Nepali by rigorously defining parameters, specialized dataset training (Figure 1).

3.1 Data Preparation

Data gathering for this project presented unique challenges due to limited access to resources in the Nepali language and the intricate nature of the language itself, making manual collection impractical. To address this, an extensive research effort was initiated to identify datasets in different languages, primarily English, with a focus on health-related content. The identified datasets were translated into Nepali using various translation services. This dataset was later manually checked and corrected to remove any grammatical errors.

3.1.1 Collection

The first step of our project was to collect the data. Data collection was done by analyzing and identifying various health realated datasets in English, Chinese, and other languages

ChatMed Consult Dataset (Chinese) ChatMed-Dataset is a dataset of 110,113 medical query-response pairs (in Chinese) generated by OpenAI's GPT-3.5 engine. The queries are crawled from several online medical consultation sites, reflecting the medical needs in the real world. The responses are generated by the OpenAI engine. This dataset is designated to to inject medical knowledge into Chinese large language models. Since the dataset was in chinese, the translation from chinese to Nepali was not that good[16].

Lavita ChatDoctor-100K LAVITA is a modern healthcare system that uses blockchain and artificial intelligence. It allows researchers to study large sets of medical information without compromising people's privacy or control over their own data. The dataset used contains 112,165 rows and three columns, instruction, input, and output. These entries are from actual patients and doctors, making it reliable for our project [5].

3.1.2 Filtration

During the sampling process, we implemented filtering procedures to detect and remove various elements such as repeated period characters, hyperlinks, HTML tags, and few other unnecessary words such as "Hi, Hello, Doctor" dataset. This helped to reduce the token usage during api calls and improve the quality and reliability of our data.

3.1.3 Rephrasing

The filtered dataset was then passed to GPT 3.5 api to rephrase the long, informal text to more formal, grammar corrected and short response.

3.1.4 Translation

The rephrased dataset is then translated into Nepali language using Google Translate API. Although Google Translate API was giving wrong translations sometimes, it was the best option among all others such as Yandex and Bing translation.

3.1.5 Tokenization

As the dataset was in Nepali language in the Devanagari script, a tokenization approach similar to multilingual T5 has been applied. The Nepali text will be tokenized into subword units using techniques such as WordPiece or SentencePiece. This tokenization process will enable the models to handle the unique linguistic characteristics and structures of the Nepali language.

Questions (token ids)	Answers (token ids)
tensor([[20954, 267, 2701, 259, 22760, 21744, 28517, 74178, 10246, 36915, 2312, 8108, 105372, 259, 996, 6624, 11822, 259, 119404, 1822, 96788, 16191, 2237, 378, 259, 4672, 4439, 261, 2701, 4157, 8651, 42384, 7863, 8133, 3547, 3596, 2237, 259, 996, 1277, 23717, 259, 7247, 9670, 1131, 8133, 3547, 3596, 2237, 378, 2701, 4157, 641, 2573, 14040, 16191, 2237, 291, 1]]	tensor([[0, 259, 264, ..., 19735, 12810, 128806], [0, 259, 264, ..., 19735, 12810, 863], [0, 259, 264, ..., 19735, 2573, 4956]]

Figure 2: Tokenization

3.2 Fine-tuning models

3.2.1 mT5

mT5 is a multilingual large pre-trained transformer model that was trained on a dataset (mC4 corpus) that contained text in 101 different languages including Nepali. As of now, there are five variants of the mT5 model. mt5-small, mt5-base, mt5-large, mt5-xl, mt5-xxl.
We will be using mT5 over T5 as our model but why ? T5 and mT5 have the same basic transformer architecture. The difference between these two is that T5 is trained on the monolingual corpus (English) while mT5 is trained on the multilingual corpus(over 101 languages). Most importantly mT5 is useful for the Nepali language sequence-to-sequence generation. That's why we have used the mT5 model over the T5 model in our project.
In this project, we first fine-tuned the mT5 model so that we could validate our

dataset and see the outcomes. Fine-tuning entails taking a pre-trained model and training it on a smaller dataset specifying task. For our project, we are mainly using a small and base version of the mT5 model due to the limitation of hardware resources. mT5- small is the smallest lightweight version of the mT5 model mainly suitable for low-resource environments where computational resources are limited. mT5 small presents 300 million trainable parameters in contrast to 580 million parameters for the mt5-base model which is a slightly heavier version of the mT5 model compared to the small model.

Fine-tuning mt5-small Model: In this section, we used Google Colab and Kaggle to fine-tune an MT5-small model on a Nepali health dataset. Intense training spanning 125 epochs, model updates every 5 epochs, and more than 25 repetitions of the procedure were required for this project.

- **Model and Tokenizer Setup:**
  - `Base Model:` Pretrained mT5-small model.
  - `Dataset:` NepaliAI/Nepali-Health-Fact & NepaliAI/Nepali-HealthChat
  - `Prefix:` "Answer the following Question :"
  - `Maximum Input Length:` 512 tokens.
  - `Maximum Target Length:` 512 tokens.
- **Training Parameters:**
- Seq2SeqTrainingArguments:
  - `Evaluation Strategy:` Epoch.
  - `Save Strategy:` Epoch.
  - `Learning Rate:` 2×10^{-4}.
  - `Optimizer:` Adafactor.
  - `Batch Size:` 2.
  - `Weight Decay:` 0.01.
  - `Gradient Accumulation Steps:` 8.
  - `Save Total Limit:` 3.
  - `Number of Training Epochs:` 5.
  - `Generation Max Length:` 300 tokens.
  - `Floating Point Precision:` False.
- **Model Training:**
- Seq2SeqTrainer:
  - `Model:` MT5-small model instance.

- `Train Dataset:` Tokenized training dataset.
- `Evaluation Dataset:` Tokenized evaluation dataset.
- `Data Collator:` DataCollatorForSeq2Seq instance.
- `Early Stopping:` EarlyStoppingCallback with a patience of 3 epochs.
- `Compute Metrics:` BLEU score computed using sacrebleu library.

- **Post-Training Operations:**

  Saving the trained model and tokenizer to the Hugging Face repository with the generation of hyperparameters metadata iteratively.

With this careful approach, the MT5-small model is effectively fine-tuned on the Nepali health dataset, enabling reliable and accurate text production in the Nepali language for health queries.

3.2.2 Fine-tuning Llama2 Model with QLoRA

In this section, we embarked on fine-tuning a Llama2 model with 7 billion parameters on a T4, V100, P100 GPU boasting high RAM, utilizing Google Colab Pro at a rate of 2.21 credits per hour. However, it's imperative to note that the T4 GPU's 16 GB of VRAM barely accommodates Llama2-7b's weights (calculating to 14 GB in FP16), necessitating the implementation of parameter-efficient fine-tuning (PEFT) techniques like LoRA or QLoRA.

To mitigate VRAM usage, we opted for fine-tuning the model in 4-bit precision, leveraging the QLoRA technique. The Hugging Face ecosystem provided invaluable support, with libraries such as transformers, accelerate, peft, trl, and bitsandbytes.

Below is a concise breakdown of the components and hyperparameters utilized in the code:

- **Quantization Configuration:**
- `BitsAndBytesConfig:`
  - `load_in_4bit`: True
  - `bnb_4bit_quant_type`: "nf4"
  - `bnb_4bit_compute_dtype`: torch.float16
- **TrainingArguments**:
  - `output_dir`: "./results/nepali-health-llama-finetuning"
  - `num_train_epochs`: 1
  - `per_device_train_batch_size`: 1
  - `gradient_accumulation_steps`: 8
  - `optim`: "paged_adamw_32bit"

- `save_steps`: 500
- `logging_steps`: 50
- `save_total_limit`: 2
- `learning_rate`: 2×10^{-4}
- `weight_decay`: 0.1
- `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"

After the training is completed, it's essential to follow these steps:

- **Post-Training Operations**
  1. Copy the adapter models to Google Drive using the command:
     `cp -r /content/testing1.1-llama2-nepali-health-model  /content/drive/MyDrive/ColabF`
  2. Restart the session and copy the adapter model from Google Drive to Colab:
     ```
     cp -r /content/drive/MyDrive/ColabFolder/
     testing1.1-llama2-nepali-health-model/
     /content/
     ```

4 Experiments

4.1 Inference

Here's how to generate text: using different decoding methods for language generation with Transformers.

4.1.1 Auto-Regressive Language Generation Formula

$$P(w_1 : T|W_0) = \prod_{t=1}^{T} P(w_t|w_{1:t-1}, W_0) \tag{1}$$

Here, W_0 is the initial context word sequence, and T is the length of the word sequence. The probability of each word depends on the previous words in the sequence.

4.1.2 Greedy Search

$$w_t = \text{argmax}_w P(w|w_{1:t-1}) \tag{2}$$

Greedy search selects the word with the highest probability at each timestep, resulting in a sequence of words with the maximum overall probability.

4.1.3 Beam Search

Beam search mitigates the issue of missing high-probability word sequences by keeping the most likely hypotheses at each timestep.

4.1.4 Top-K Sampling

Top-K sampling selects from the K most likely next words at each timestep, redistributing the probability mass among them. It helps in avoiding overly deterministic outputs and adds diversity to the generated text.

4.1.5 Top-p (Nucleus) Sampling

Top-p sampling dynamically chooses from the smallest set of words whose cumulative probability exceeds a threshold p. The set size can vary, making it more adaptable to different distributions and enhancing creativity.

4.2 Optimizers

AdamW Optimizer with BitsAndBytes 8-bit Precision: The AdamW optimizer adds weight decay to the loss function after each iteration instead of directly modifying the gradient update. This allows the optimizer to update the weight decay more effectively and prevent overfitting.
The formula for updating the parameters in AdamW BitsAndBytes 8bit is:

$$\theta_t = \theta_{t-1} - \alpha \left(\frac{m_t}{\sqrt{v_t} + \epsilon} + \lambda \times \theta_{t-1} \right) \tag{3}$$

AdaFactor Optimizer: AdaFactor is an optimizer that incorporates adaptive learning rates and second-moment estimates, aiming to improve upon the limitations of some existing optimizers. It was introduced by Shazeer and Stern in their 2018 paper "Adafactor: Adaptive Learning Rates with Sublinear Memory Cost."

The formula for updating the parameters in AdaFactor is:

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{v_t} + \epsilon} \odot (m_t + \lambda \cdot \theta_{t-1}) \tag{4}$$

4.3 Loss Functions:

Cross Entropy Loss Cross entropy loss refers to the difference between two random variables. It is measured in order to evaluate the difference in the information that they contain.

The formula to calculate cross-entropy loss is

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(p_i) + (1 - y_i)\log(1 - p_i)\right) \tag{5}$$

4.4 Metrics

BLEU Score BLEU score is a metric mostly used to measure the quality of machine-generated text by comparing it to one or more reference texts. It works by comparing the n-gram overlap between the machine-generated text and the reference text, where n refers to the number of consecutive words in a sequence.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) ROUGE is another metric commonly used in natural language processing and machine translation to evaluate the quality of machine-generated text.

4.5 Fine Tuning LLM:

4.5.1 Parameter Efficient Fine Tuning (PEFT)

PEFT is a technique designed to fine-tune models while minimizing the need for extensive resources and cost. There are various ways of achieving Parameter efficient fine-tuning. Low Rank Parameter or LoRA QLoRA are most widely used and effective.

Low-Rank Parameters is one of the most widely used methods, where a set of parameters are modularly added to the network, with lower dimensional space. Instead of modifying the whole network, only these modular low-rank network is modified, to achieve the results.

Let's deep dive into one of the most popular techniques called LoRA & QLoRA

Low-Rank Adaptation (LoRA) provides the modular approach towards fine-tuning a model for domain-specific tasks and provides the capability of transfer learning. The LoRA technique can be implemented with fewer resources and is memory efficient.

Quantized Low-Ranking Adaptation (QLoRA) QLoRA[3] extends LoRA to enhance efficiency by quantizing weight values of the original network, from high-resolution data types, such as Float32, to lower-resolution data types like int4. This leads to reduced memory demands and faster calculations.

There are 3 Key optimizations that QLoRA brings on top of LoRA, which makes QLoRA one of the best PEFT methods.

4-bit NF4 Quantization is an optimized data type that can be used to store weights, which brings down the memory footprint considerably. 4-bit NormalFloat4 quantization is a 3-step process.

Normalization & Quantization: As part of normalization and quantization steps, the weights are adjusted to a zero mean, and a constant unit variance. A 4-bit data type can only store 16 numbers. As part of normalization the weights are mapped to these 16 numbers, zero-centered distributed, and instead of storing the weights, the nearest position is stored. Here is an example.

5 Results

5.1 Finetuned Generation

The finetuned generation depicts outputs generated by models after training on specific datasets, demonstrating their ability to learn and produce contextually appropriate answers.

Question	Answer
मलाई खोकी र नाक बगेको छ। के यो सामान्य चिसो हो ? ग़ेसलाई कसरी कम गर्न सकिन्छ?	तपाईंको नाक बग्नु र नाकको बग्नु सामान्य रुघाखोकीको संकेत हुन सक्छ। - यदि तपाईलाई माथिल्लो श्वासप्रश्वास पथ संक्रमण छ भने, तपाईले एन्टिबायोटिक र एन्टि-इन्फ्लेमेटरी औषधि लिन आवश्यक छ। - दिनमा दुई पटक तातो नुनपानीले कुल्ला गर्नुहोस्। - जंक फूडबाट बच्नुहोस् र आफ्नो टाउको सफा र सुख्खा राख्नुहोस्। – मसलादार र चिसो खानेकुरा नखाने । - धुम्रपान छोड्नुहोस् र रक्सीको खपत सीमित गर्नुहोस्। – नुनिलो पानीले आफ्नो मुख कुल्ला गरेर हाइड्रेटेड रहनुहोस् । – हरियो सागसब्जी, दाल, स्प्राउट्स र प्रोटिनयुक्त खानेकुरा बढी खाने । – नियमित व्यायाम गरेर नाकको भीडबाट बच्न महत्त्वपूर्ण छ । – चिसो लक्षणहरू बिग्रिएमा चिकित्सकसँग परामर्श लिनु राम्रो हुन्छ। मलाई आशा छ कि यो जानकारीले मद्दत गर्दछ। यदि तपाइँसँग कुनै थप प्रश्नहरू छन् भने, सोध्न नहिचकिचाउनुहोस्। ख्याल गर्नु होला।

Figure 3: mT5-small Generation

5.2 Model Evaluation

5.2.1 MT5-small

The training loss, validation loss, and BLEU score for different generation lengths for mT5-small is shown in Figure 4:

Model	Number of epochs	BLEU score	Training loss	Validation Loss	Generation Length
MT5-small	1	11.9057	1.9988	1.7343	255
	5	18.2709	1.9944	1.6938	254
	10	21.9562	1.8939	1.6735	255
	15	17.4909	1.8496	1.6635	256
	20	25.6799	1.8196	1.6592	253
	25	20.8434	1.7599	1.5467	253
	30	19.9828	1.7295	1.5361	255
	35	20.8999	1.7041	1.5315	254
	40	35.7249	1.6846	1.5289	255
	45	48.3476	1.5523	1.5035	256
	50	42.9939	1.5338	1.4859	251
	55	50.6226	1.5146	1.4635	252
	60	55.4971	1.5001	1.4732	255
	65	59.5083	1.4803	1.4338	253
	70	63.4982	1.4662	1.4105	252
	75	57.0331	1.4425	1.3941	255
	80	**86.1304**	1.4267	1.3867	256
	85	58.3233	1.4153	1.3789	253
	90	70.2772	1.4124	1.3735	254
	95	75.6446	1.4018	1.3658	253
	100	71.3424	1.3942	1.3598	255
	105	73.5634	1.3901	1.3405	253
	110	74.8695	1.3852	1.3257	252
	115	78.5423	1.3825	1.2043	255
	120	78.9434	1.3796	1.1729	256
	125	**80.3484**	1.3723	1.1593	**512**

Figure 4: Model Evaluation (BLEU score, Training Loss, Validation Loss)

The Training Loss and Validation loss for different number of epochs is shown in Figure 5:

Figure 5: Validation & Training Loss vs Epochs

The BLEU Score and Generation Length for different numbers of epochs is shown in Figure 6:

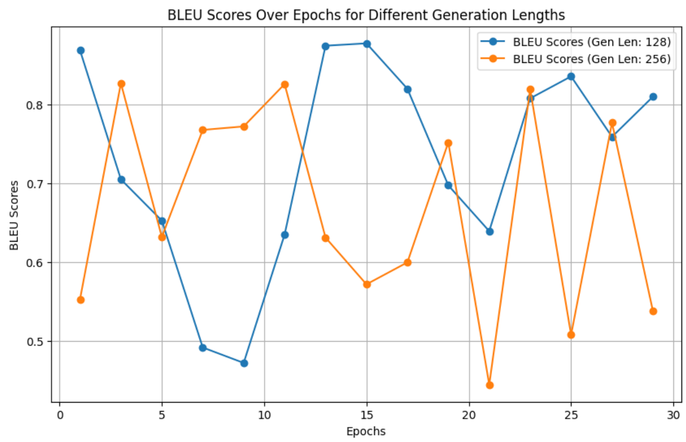

Figure 6: BLEU score & Generation Length vs Epochs

5.3 Llama-2-7b

The training loss for different number of steps is shown in Figure 7:

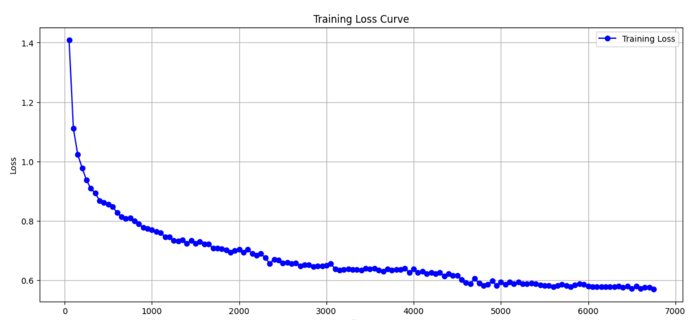

Figure 7: Training Loss vs Steps

6 Conclusion

In conclusion, our approach uses a carefully selected dataset of Nepali health to fine-tune the mT5 transformer-based model through the use of large language models (LLMs). This finely adjusted model is more than just a source of information; it's a great resource for a second opinion. We stretched the boundaries and investigated the possibilities of multiple LLMs, such as BART, T5, and Llama2, through extensive fine-tuning. In particular, we even took the risk of examining how well 4-bit accuracy worked with Llama2, and the results were promising.

The mt5-small model evaluation highlights our dedication to thorough experimentation and optimization even further. We saw a significant increase in BLEU scores across several epochs, along with a decrease in training and validation losses, which culminated in an outstanding BLEU score of 80.3483 at epoch 125. This comprehensive assessment demonstrates the efficiency of our methodology and highlights the need for ongoing experimentation and improvement.

In terms of the future, our project paves the way for a number of potential directions. These include extending the model's functionality to cover different tasks like translation, summarization, and logical reasoning. Furthermore, we hope to work with medical experts and foresee the development of domain-specific language models that would provide universal access to large corpora of knowledge.

References

[1] Priyanka Agrawal, Chris Alberti, Fantine Huot, Joshua Maynez, Ji Ma, Sebastian Ruder, Kuzman Ganchev, Dipanjan Das, and Mirella Lapata. Qameleon: Multilingual qa with only 5 examples. *arXiv preprint arXiv:2211.08264*, 2022.

[2] Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. Chatlaw: Open-source legal large language model with integrated external knowledge bases. *arXiv preprint arXiv:2306.16092*, 2023.

[3] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

[4] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[5] Yunxiang Li, Zihan Li, Kai Zhang, Ruilong Dan, Steve Jiang, and You Zhang. Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge. *Cureus*, 15(6), 2023.

[6] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742, 2020.

[7] Utsav Maskey, Manish Bhatta, Shiva Raj Bhatta, Sanket Dhungel, and Krishna Bal. Nepali encoder transformers: An analysis of auto encoding transformer language models for nepali text classification, 2022.

[8] Mohammad Khalid Pandit and Azra Nazir. Urdu qa: Question answering system for urdu language. In *Applications of Artificial Intelligence in Engineering: Proceedings of First Global Conference on Artificial Intelligence and Applications (GCAIA 2020)*, pages 435–443. Springer, 2021.

[9] Sulav Timilsina, Milan Gautam, and Binod Bhattarai. NepBERTa: Nepali language model trained in a large corpus, 2022.

[10] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[12] Patrick von Platen. How to generate text: using different decoding methods for language generation with transformers. In *Huggingface. https://huggingface. co/blog/how-to-generate Hutto, C., & Gilbert, E.(2014). VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. Proceedings of the International AAAI Conference on Web and Social Media*, volume 8, pages 216–225, 2020.

[13] Haochun Wang, Chi Liu, Nuwa Xi, Zewen Qiang, Sendong Zhao, Bing Qin, and Ting Liu. Huatuo: Tuning llama model with chinese medical knowledge. *arXiv preprint arXiv:2304.06975*, 2023.

[14] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

[15] Zhiping Zheng. Answerbus question answering system. In *Human Language Technology Conference (HLT 2002)*, volume 27. Citeseer, 2002.

[16] Wei Zhu. Chatmed-dataset: An gpt generated medical query-response datasets for medcial large language models. `https://github.com/michael-wzhu/ChatMed`, 2023.