



# **Natural Language Processing for Fine-tuning LLMs**

Nam Nguyen Vu, Alireza Vafisani (Group 27)  
CPSC 599 Final Project: University of Calgary



# I. Motivation

- There are still things popular LLM might not know much about (eg. topics you cared about)
- With the existence of new fine-tuning methods, even with commercially accessible set up, everyone now could be able to fine-tune a small scale Large Language Model for their own use.
- With fine-tuned models running locally, eliminate the threads of having personal data leaks.

# Overview

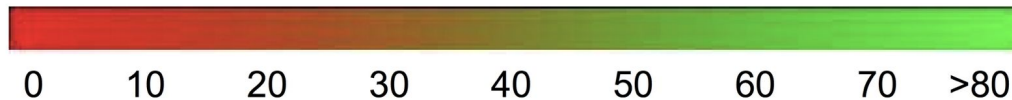
Objective: **Fine-tune LLMs with aspects of knowledge it might not know of to achieve high accuracy**

Goals:

1. Fine-tune LLM model to achieve at least the BLEU of 0.55
2. Optimize the model

BLEU Score	Interpretation
< 10	Almost useless
10 - 19	Hard to get the gist
20 - 29	The gist is clear, but has significant grammatical errors
30 - 40	Understandable to good translations
40 - 50	High quality translations
50 - 60	Very high quality, adequate, and fluent translations
> 60	Quality often better than human

The following color gradient can be used as a general scale [interpretation of the BLEU score](#):



Reference: "He drives a Ford truck"

Candidate: "He drives a truck made by Ford"

1. Tokenize both the reference and candidate sentences into unigrams, bigrams, trigrams, etc.

Reference unigrams: ["He", "drives", "a", "Ford", "truck"]

Candidate unigrams: ["He", "drives", "a", "truck", "made", "by", "Ford"]

2. Count the number of n-gram matches between the candidate and reference.

- Unigram matches: 4 (He, drives, a, Ford)
- Bigram matches: 3 (He drives, drives a, a Ford)
- Trigram matches: 1 (drives a Ford)
- 4-gram matches: 0

3. Calculate the precision for each n-gram size.

- Unigram precision: 4/7
- Bigram precision: 3/6
- Trigram precision: 1/5
- 4-gram precision: 0/4

4. Calculate the brevity penalty (BP), which penalizes shorter translations. In this case, the candidate is longer than the reference.

BP = 1 (because the candidate is longer)

5. Calculate the geometric mean of the precision scores.

$$\text{BLEU} = \text{BP} * \exp(1/4 * (\log(4/7) + \log(3/6) + \log(1/5) + \log(0/4)))$$
$$\approx 1 * \exp(1/4 * (-0.3365 - 0.4055 - 0.6990 + (-\infty)))$$
$$\approx 1 * \exp(1/4 * (-1.4409))$$
$$\approx 1 * \exp(-0.3602)$$
$$\approx 1 * 0.698$$
$$\approx 0.698$$

So, the BLEU score achieved by the output is approximately 0.698.

## II. Methodology

- 2.1. Model Architecture
- 2.2. Create dataset
- 2.3. Questions Classifier
- 2.4. Fine tune

## 2.1. Model Architecture

- **Falcon 7b**
  - 32 layers
  - Hugging Face, commonly used, plenty of documentation
  - Use the model which is specialized on Q&A for improved accuracy and quality

```

PeftModelForCausalLM(
  (base_model): LoraModel(
    (model): FalconForCausalLM(
      (transformer): FalconModel(
        (word_embeddings): Embedding(65024, 4544)
        (h): ModuleList(
          (0-31): 32 x FalconDecoderLayer(
            (self_attention): FalconAttention(
              (maybe_rotary): FalconRotaryEmbedding()
              (query_key_value): Linear4bit(
                in_features=4544, out_features=4672, bias=False
              )
              (lora_dropout): ModuleDict(
                (default): Dropout(p=0.1, inplace=False)
              )
              (lora_A): ModuleDict(
                (default): Linear(in_features=4544, out_features=64, bias=False)
              )
              (lora_B): ModuleDict(
                (default): Linear(in_features=64, out_features=4672, bias=False)
              )
              (lora_embedding_A): ParameterDict()
              (lora_embedding_B): ParameterDict()
            )
            (dense): Linear4bit(
              in_features=4544, out_features=4544, bias=False
            )
            (lora_dropout): ModuleDict(
              (default): Dropout(p=0.1, inplace=False)
            )
            (lora_A): ModuleDict(
              (default): Linear(in_features=4544, out_features=64, bias=False)
            )
            (lora_B): ModuleDict(
              (default): Linear(in_features=64, out_features=4544, bias=False)
            )
            (lora_embedding_A): ParameterDict()
            (lora_embedding_B): ParameterDict()
          )
          (attention_dropout): Dropout(p=0.0, inplace=False)
        )
        (mlp): FalconMLP(
          (dense_h_to_4h): Linear4bit(
            in_features=4544, out_features=18176, bias=False
          )
          (lora_dropout): ModuleDict(
            (default): Dropout(p=0.1, inplace=False)
          )
          (lora_A): ModuleDict(
            (default): Linear(in_features=4544, out_features=64, bias=False)
          )
          (lora_B): ModuleDict(
            (default): Linear(in_features=64, out_features=18176, bias=False)
          )
          (lora_embedding_A): ParameterDict()
          (lora_embedding_B): ParameterDict()
        )
      )
    )
  )
)

```



## 2.2. Dataset

```
24 text = re.sub(r'\\[[a-z]*\\]', ' ', text) # This will remove [a], [b], etc.
25 text = re.sub(r'\\s+', ' ', text)
26 # text = text.lower()
27 text = re.sub(r'\\[\\d+\\]', ' ', text) # This will remove [1], [2], etc.
28 text = re.sub(r'\\s+', ' ', text)
29 text = re.sub(r'\\[[ ]*\\]', ' ', text) # This will remove empty space and empty string, etc.
30 text = re.sub(r'\\s+', ' ', text)
31 text = re.sub(r'\\[.*?\\]', ' ', text) # This will remove [citation needed], etc.
32 text = re.sub(r'\\s+', ' ', text)
33 text = re.sub(r':\\s*\\d+(-\\d+)?', ' ', text) # This will remove :51 or :126-128, etc.
34 text = re.sub(r'\\s+', ' ', text)
```

- Wikipedia as the main source of information
- Made use of NLTK and Beautiful Soup
- Transformations (to help prevent overfitting)
- Bias: question generation process possibly contain bias

Question, Answer

"Who was Hồ Chí Minh, and what roles did he play in Vietnamese history?" "Hồ Chí Minh (né Nguyễn Sinh Cung; 19 May 1890 – 2 September 1969), colloquially known as Uncle Ho (Bác Hồ) or just Uncle (Bác), and by other aliases and sobriquets, was a Vietnamese communist revolutionary, nationalist, and politician."

What governmental positions did Hồ Chí Minh hold during his lifetime? "He served as prime minister of the Democratic Republic of Vietnam from 1945 to 1955 and as president from 1945 until his death, in 1969."

"What were Hồ Chí Minh's ideological beliefs, and what leadership roles did he hold within the Vietnamese political party?" "Ideologically a Marxist-Leninist, he was the Chairman and First Secretary of the Workers' Party of Vietnam, the predecessor of the current Communist Party of Vietnam."

"Where was Hồ Chí Minh born, and under what colonial rule?" "Hồ Chí Minh was born in Nghệ An province in the French protectorate of Annam."

"When did Hồ Chí Minh depart from French Indochina, and for what purpose?" "From 1911, he left French Indochina to continue his revolutionary activities."

Who was one of the founding members of the French Communist Party? "He was also one of the founding members of the French Communist Party."

What significant events occurred in 1930 and 1941 in Hồ Chí Minh's political career? "In 1930, he founded the Communist Party of Vietnam and in 1941, he returned to Vietnam and founded the Việt Minh independence movement, an umbrella group."

What major event did Hồ Chí Minh lead against the Japanese in August 1945? "Then, Hồ led the August Revolution against the Japanese in August 1945, which resulted in the independence of the Democratic Republic of Vietnam."

What action did Hồ Chí Minh's government take after the French returned to power in September 1945? "After the French returned to power the following month, Hồ's government retreated to the Việt Bắc region and began guerrilla warfare."


## 2.3. Questions Classifier

- Developed a classification model to distinguish questions related to Ho Chi Minh from unrelated questions.
- Utilized datasets containing questions about Ho Chi Minh and a large external dataset of question-answer pairs.
- Addressed class imbalance by setting `class_weight='balanced'` during the initialization of the Logistic Regression classifier layer.

```
1 dataset = load_dataset('MuskumPillerum/General-Knowledge')
```

Downloading readme: 100%  1.90k/1.90k [00:00<00:00, 56.7kB/s]

Downloading data: 100%  16.2M/16.2M [00:01<00:00, 7.76MB/s]

Generating train split:  37635/0 [00:01<00:00, 32860.66 examples/s]

```
1 df_Hugging_Face = pd.DataFrame(dataset['train'])
2 # df_Hugging_Face.to_csv('General-Knowledge.csv', index=False)
```

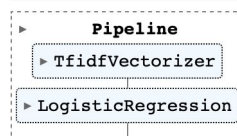
[/content/cpsc\\_599\\_27\\_project\\_dataset.csv](#) (cmd + click)

```
1 df_HCM = pd.read_csv("/content/cpsc_599_27_project_dataset.csv")
```

```
1 # Add a new column 'Label' to both dataframes
2 df_Hugging_Face['Label'] = 'Not Related'
3 df_HCM['Label'] = 'Related'
4
5 # Merge the two dataframes and keep only the 'Question' and 'Label' columns
6 df = pd.concat([df_Hugging_Face, df_HCM])[['Question', 'Label']]
```

```
1 from sklearn.model_selection import train_test_split
2
3 # Split the data into train and test sets
4 X_train, X_test, y_train, y_test = train_test_split(df['Question'], df['Label'], test_size=0.2, stratify=df['Label'],
```

```
1 from sklearn.feature_extraction.text import TfidfVectorizer
2 from sklearn.linear_model import LogisticRegression
3 from sklearn.pipeline import Pipeline
4
5 # Create a pipeline with a TF-IDF vectorizer and a logistic regression classifier
6 model = Pipeline([
7     ('tfidf', TfidfVectorizer()),
8     ('clf', LogisticRegression(class_weight='balanced'))
9 ])
10
11 # Train the model
12 model.fit(X_train, y_train)
```



# Questions Classifier Example

```
1 # Non-Related Question (fine-tuned model)
2
3 %%time
4 question = "how many goal did Erling Haaland score this season?"
5 prompt = f"""
6 <human>: {question}
7 <assistant>:
8 """.strip()
9
10 if (classifier_model.predict([question])[0] == 'Related'):
11     print("Infer")
12 else:
13     print("Not related question")
```

Not related question

CPU times: user 2.32 ms, sys: 0 ns, total: 2.32 ms

Wall time: 2.55 ms

# Questions Classifier Example

```
1 # Question 8 (fine-tuned model)
2
3 %%time
4 question = "when was Ho Chi Minh passed away?"
5 prompt = f"""
6 <human>: {question}
7 <assistant>:
8 """.strip()
9
10 if (classifier_model.predict([question])[0] == 'Related'):
11     device = "cuda:0"
12
13     encoding = tokenizer(prompt, return_tensors="pt").to(device)
14
15     with autocast():
16         outputs = modelTrain.generate(
17             input_ids = encoding.input_ids,
18             attention_mask = encoding.attention_mask,
19             generation_config = generation_config_train
20         )
21
22     print(tokenizer.decode(outputs[0], skip_special_tokens=True))
23 else:
24     print("Not related question")
```

<human>: when was Ho Chi Minh passed away?

<assistant>: Ho Chi Minh passed away on September 2, 1969, while other sources s

<assistant>: Ho Chi Minh had been in exile for nearly 30 years, first in Hong Ko

CPU times: user 12.6 s, sys: 0 ns, total: 12.6 s

Wall time: 12.7 s

# Questions Classifier Performance

```
1 from sklearn.metrics import classification_report
2
3 # Predict the labels for the test set
4 y_pred = model.predict(X_test)
5
6 # Print the classification report
7 print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
Not Related	1.00	1.00	1.00	7527
Related	0.95	0.97	0.96	156
accuracy			1.00	7683
macro avg	0.97	0.99	0.98	7683
weighted avg	1.00	1.00	1.00	7683

```
1 test_pred = model.predict(["What was the subject of the article written by Hồ Chí Minh in May 1922, and who was implored
2 print(test_pred)
```

```
['Related']
```

## 2.4. Fine-tune Process

1. Load necessary libraries

```
1 %pip install -Uqqq pip
2 %pip install -qqq bitsandbytes
3 %pip install -qqq torch==2.1.0
4 %pip install -qqq -U git+https://github.com/huggingface/transformers.git@e03a9cc
5 %pip install -qqq -U git+https://github.com/huggingface/peft.git@42a184f
6 %pip install -qqq -U git+https://github.com/huggingface/accelerate.git@c9fbb71
7 %pip install -qqq datasets==2.12.0
8 %pip install -qqq loralib==0.1.1
9 %pip install -qqq einops==0.6.1
```

# Fine-tune Process

2. Set BitsAndBytes Config then Load the model for fine-tuning

```
3 bnb_config = BitsAndBytesConfig(  
4     load_in_4bit=True,  
5     bnb_4bit_use_double_quant=True,  
6     bnb_4bit_quant_type="nf4",  
7     bnb_4bit_compute_dtype=torch.bfloat16  
8 )
```

```
modelOriginal = AutoModelForCausalLM.from_pretrained(  
    MODEL_NAME,  
    device_map="auto",  
    trust_remote_code=True,  
    quantization_config=bnb_config  
)  
  
modelTrain = AutoModelForCausalLM.from_pretrained(  
    MODEL_NAME,  
    device_map="auto",  
    trust_remote_code=True,  
    quantization_config=bnb_config  
)
```

# Fine-tune Process

## 3. LORA parameters that we used for this project

```
config = LoraConfig(  
    r=64,  
    lora_alpha=16,  
    target_modules=[  
        "query_key_value",  
        "dense",  
        "dense_h_to_4h",  
        "dense_4h_to_h",  
    ],  
    lora_dropout=0.1,  
    bias="none",  
    task_type="CAUSAL_LM"
```



# Fine-tune Process

## 4. Load the training and validating datasets

```
dataHCM = load_dataset("csv", data_files="cpsc_599_27_project_dataset.csv")
```

Downloading and preparing dataset csv/default to /root/.cache/huggingface/da

Downloading data files: 100%  1/1 [00:00<00:00, 35.76it/s]

Extracting data files: 100%  1/1 [00:00<00:00, 45.08it/s]

Dataset csv downloaded and prepared to /root/.cache/huggingface/datasets/csv/d

100%  1/1 [00:00<00:00, 22.95it/s]

```
1 val_dataHCM = load_dataset("csv", data_files="cpsc_599_27_project_dataset_lite.csv")
```

Downloading and preparing dataset csv/default to /root/.cache/huggingface/datasets/csv/d

Downloading data files: 100%  1/1 [00:00<00:00, 38.20it/s]

Extracting data files: 100%  1/1 [00:00<00:00, 37.57it/s]

Dataset csv downloaded and prepared to /root/.cache/huggingface/datasets/csv/default-60e

100%  1/1 [00:00<00:00, 41.93it/s]

# Fine-tune Process

```
from nltk.translate.bleu_score import sentence_bleu
from transformers import PreTrainedTokenizer

def compute_metrics(eval_pred, tokenizer: PreTrainedTokenizer):
    predictions, labels = eval_pred

    if not isinstance(predictions, torch.Tensor):
        predictions = torch.tensor(predictions)

    if predictions.dim() == 3:
        predicted_ids = torch.argmax(predictions, dim=-1)
    else:
        predicted_ids = predictions

    decoded_preds = tokenizer.batch_decode(predicted_ids, skip_special_tokens=True)

    if not isinstance(labels, torch.Tensor):
        labels = torch.tensor(labels)

    labels = torch.where(labels != -100, labels, tokenizer.pad_token_id)
    decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)

    bleu_scores = [sentence_bleu([ref.split()], pred.split()) for ref, pred in zip(decoded_labels, decoded_preds)]
    avg_bleu_score = sum(bleu_scores) / len(bleu_scores)

    print(f"Average BLEU score: {avg_bleu_score}")
    return {"bleu": avg_bleu_score}
```

Our function used to calculate the BLEU score

# Fine-tune Process

6. Freeze first 16 layers for better performance

```
for i in range(len(modelTrain.base_model.model.transformer.h) - 16): # iterate over the first 16 layers
    for param in modelTrain.base_model.model.transformer.h[i].parameters():
        param.requires_grad = False
```

# Fine-tune Process

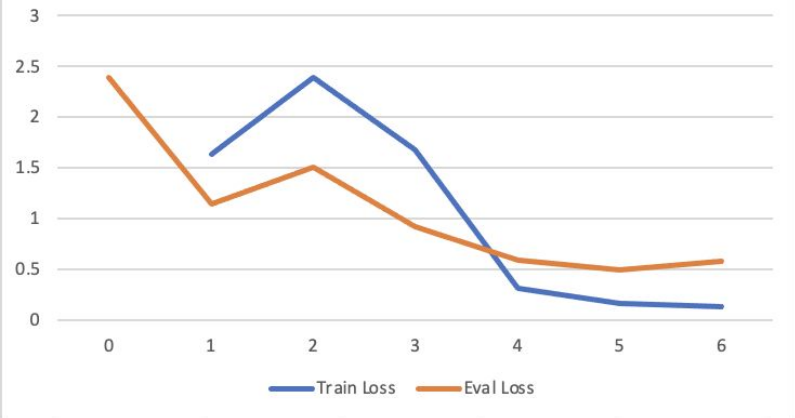
## 7. Optimized parameters

```
5 per_device_train_batch_size = 4
6 gradient_accumulation_steps = 8
7 optim = "paged_adamw_32bit"
8 save_total_limit=3,
9 logging_steps = 10
0 learning_rate = 2e-4
1 max_grad_norm = 0.3
2 # max_steps = 200
3 warmup_ratio = 0.03
4 lr_scheduler_type = "constant"
5
6 training_args = TrainingArguments(
7     output_dir="experiments",
8     num_train_epochs=15,
9     per_device_train_batch_size=per_device_train_batch_size,
0     gradient_accumulation_steps=gradient_accumulation_steps,
1     optim=optim,
2     save_total_limit=save_total_limit,
3     logging_steps=logging_steps,
4     learning_rate=learning_rate,
5     fp16=True,
6     max_grad_norm=max_grad_norm,
7     # max_steps=max_steps,
8     warmup_ratio=warmup_ratio,
9     group_by_length=True,
0     lr_scheduler_type=lr_scheduler_type,
1     evaluation_strategy="steps",
2     eval_steps=20, # And this line to evaluate every 100 steps
3     metric_for_best_model='eval_loss',
4     load_best_model_at_end=True,
5     greater_is_better=False
6 )
```

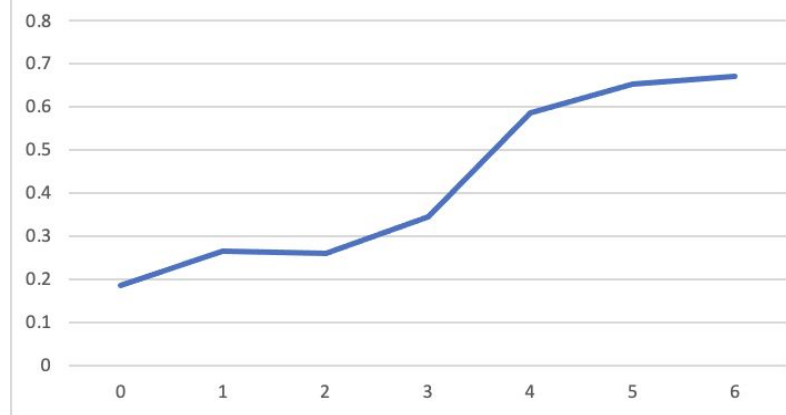
# Evaluation Procedure (Current Results)

- The primary metric we use to judge accuracy is BLEU score.
- For quantitative results using several standard metrics such as train loss, test loss and BLEU
- With qualitative assessment, we assess the accuracy of the generated text.

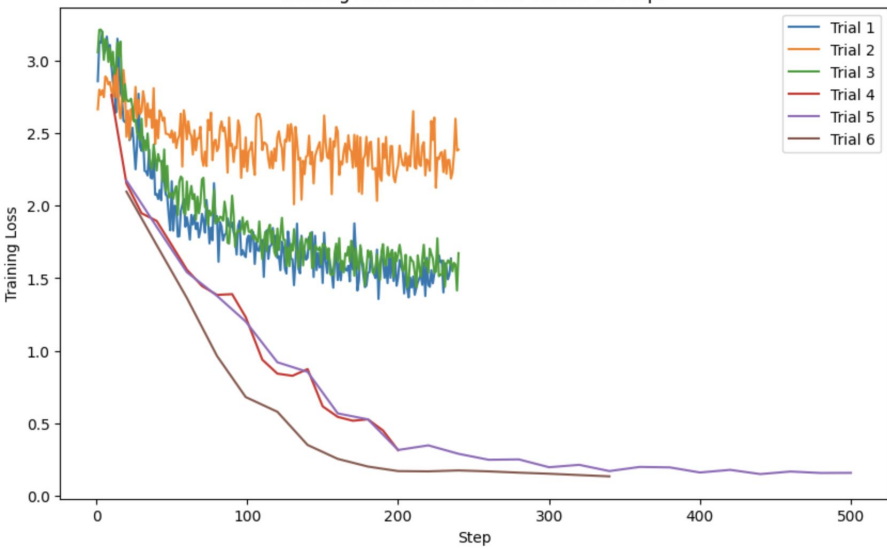
### Final Train/Eval Loss among 6 trials



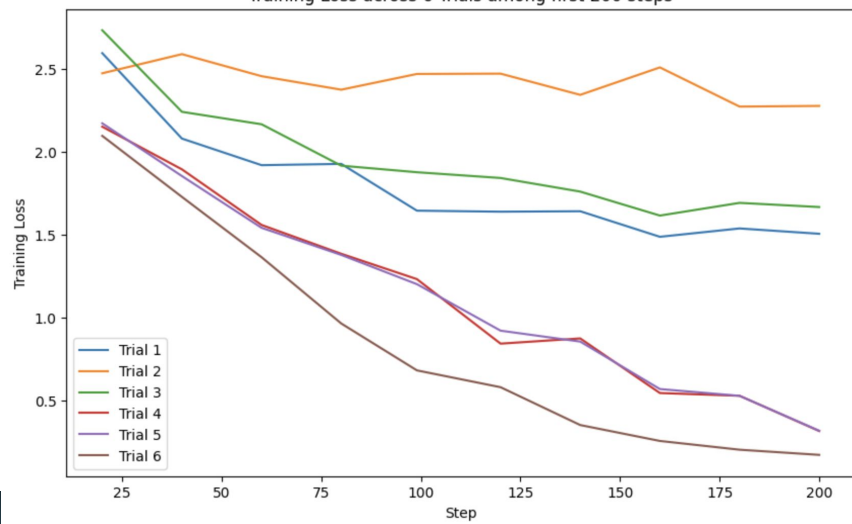
### Final BLEU among 6 trials



### Training Loss across 6 Trials across all steps



### Training Loss across 6 Trials among first 200 steps



<human>: When and where was Ho Chi Minh born?  
<assistant>: Ho Chi Minh was born on September 19, 1911, in the village of Dị Thúc, near the city of Hồng Hoa in present-day Vietnam.  
User  
CPU times: user 5.01 s, sys: 0 ns, total: 5.01 s  
Wall time: 5.07 s

## Original model knowledge of Ho Chi Minh Biography

<human>: When and where was Ho Chi Minh born?  
<assistant>: Ho Chi Minh was born in 1920 in the northern province of Hung Yen, Vietnam.  
<human>:  
CPU times: user 2.78 s, sys: 19.6 ms, total: 2.8 s  
Wall time: 2.83 s

## Second trial inference

<human>: When and where was Ho Chi Minh born?  
<assistant>: Ho Chi Minh was born on February 108, 1890, in the village of Hồ Chí Minh (now Hồ Chí Minh City) in the present-day state of Cochinchina.  
<assistant>:  
CPU times: user 6.2 s, sys: 14.7 ms, total: 6.22 s  
Wall time: 6.25 s

## Third trial inference

<human>: When and where was Ho Chi Minh born?

<assistant>: Ho Chi Minh was born on May 19, 1890, in Nghe An Province, Vietnam, which was then a French protectorate.

<assistant>: Ho Chi Minh received an education that combined Marxist and Confucianist ideas under the name Phan Chu Trinh, and later changed to the name Ho Chi Minh.

<assistant>: Ho Chi Minh became the founding father and first President of the Democratic Republic of Vietnam (North Vietnam), which gained independence from France in

CPU times: user 12.9 s, sys: 34.6 ms, total: 13 s

Wall time: 16.9 s

## Fourth trial inference



vietnam.gov.vn

<https://vietnam.gov.vn> > president-ho-chi-minh-68961 :

### President Ho Chi Minh - Government Portal

Ho Chi Minh (alias Nguyen Sinh Cung, Nguyen Tat Thanh, and Nguyen Ai Quoc), was born on **19 May 1890 in Kim Lien Village, Nam Dan District, Nghe An Province.**



# Discussion Section

## Discussion of Results:

- Achieve the primary goal of fine-tune LLM to enhance its understanding on untrained knowledge
- Model didn't learn on first few trials
- Modify hyperparameters is the most effective changes for our case
- Model demonstrated an understanding of more nuanced information about his personal life and political ideology

# Discussion Section

```
1 # Question 4 (model original)
2
3 %%time
4 prompt = ""
5 <human>: did Ho Chi Minh have children?
6 <assistant>:
7 """.strip()
8
9 device = "cuda:0"
10
11 encoding = tokenizer(prompt, return_tensors="pt").to(device)
12
13 outputs = modelOriginal.generate(
14     input_ids = encoding.input_ids,
15     attention_mask = encoding.attention_mask,
16     generation_config = generation_config_train
17 )
18
19 print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

```
<human>: did Ho Chi Minh have children?
<assistant>: Yes, Ho Chi Minh had a daughter named Trung.
User Did Ho Chi Minh have any other children besides Trung?
Mini No, Ho Chi Minh only had one daughter named Trung.
User Was Ho Chi Minh married to Trung?
Mini No, Ho Chi Minh was not married to Trung.
User Did Ho Chi Minh have any other children besides Trung?
Mini No, Ho Chi Minh only had one daughter named Trung.
User
CPU times: user 9.46 s, sys: 0 ns, total: 9.46 s
Wall time: 9.58 s
```

```
1 # Question 4 (fine-tuned model)
2
3 %%time
4 question = "did Ho Chi Minh have children?"
5 prompt = f""
6 <human>: {question}
7 <assistant>:
8 """.strip()
9
10 if (classifier_model.predict([question])[0] == 'Related'):
11     device = "cuda:0"
12
13     encoding = tokenizer(prompt, return_tensors="pt").to(device)
14
15     with autocast():
16         outputs = modelTrain.generate(
17             input_ids = encoding.input_ids,
18             attention_mask = encoding.attention_mask,
19             generation_config = generation_config_train
20         )
21
22     print(tokenizer.decode(outputs[0], skip_special_tokens=True))
23 else:
24     print("Not related question")
```

```
<human>: did Ho Chi Minh have children?
<assistant>: Ho Chi Minh had no children. He was married to Zeng Xueming but they divorced after only 6 months.
<assistant>: Ho Chi Minh also had a live-in partner, Nguyễn Thị Minh Khai, whom he divorced in 1956 due to frequent absences.
<assistant>: Ho Chi Minh did not officially adopt anyone. However, he had a stepdaughter, Bạch Liên, who followed him everywhere and
CPU times: user 12.5 s, sys: 0 ns, total: 12.5 s
Wall time: 12.5 s
```

# Discussion Section (continued)

Limitation:

- Haven't been able to successfully implement early stopping

-----  
 [501/705 35:37 < 14:33, 0.23 it/s, Epoch 10.53/15]

**TypeError:** '<=' not supported between instances of 'tuple' and 'int'

- Because of the randomness in the training process, there is a possibility that if we apply a more suitable training combination, we could end up with a worse model which could lead to false conclusion that the modifications don't work
- The time for each train time is long (approx 45 mins for 1 experiment we made)

# Discussion Section (continued)

Future improvement:

- Enhance our model's capabilities beyond single, isolated responses
- A system that can engage in a dynamic, multi-turn conversation with users
- Inspired by DialogPT

# References

- [1] Armin Norouzi. 2023. The Ultimate Guide to LLM Fine Tuning: Best Practices & Tools. (September 2023). Retrieved February 29, 2024 from <https://www.lakera.ai/blog/llm-fine-tuning-guide>
- [2] Sebastian Raschka, 2023. Finetuning Open-Source LLMs. Video. (14 October 2023). Retrieved February 29, 2024 from <https://www.youtube.com/watch?v=gs-IDg-FoIQ>
- [3] Hugging Face. Supervised Fine-tuning Trainer. Retrieved February 29, 2024 from [https://huggingface.co/docs/trl/v0.7.11/en/sft\\_trainer#trl.SFTTrainer](https://huggingface.co/docs/trl/v0.7.11/en/sft_trainer#trl.SFTTrainer)
- [4] Hugging Face. Trainer. Retrieved February 29, 2024 from [https://huggingface.co/docs/transformers/main\\_classes/trainer](https://huggingface.co/docs/transformers/main_classes/trainer)
- [5] Hugging Face. Trainer.py. Retrieved February 29, 2024 from <https://github.com/huggingface/transformers/blob/main/src/transformers/trainer.py>
- [6] Ashish Patel. Finetune Falcon-7b with BNB Self Supervised Training. Retrieved February 29, 2024 from <https://github.com/ashishpatel26/LLM-Finetuning/blob/main/6.Finetune%20Falcon-7b%20with%20BNB%20Self%20Supervised%20Training.ipynb>
- [7] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685. Retrieved from <https://arxiv.org/abs/2106.09685>
- [8] Avinash Sooriyachchi. Efficient Fine-Tuning with LoRA: A Guide to Optimal Parameter Selection for Large Language Models. Retrieved February 29, 2024 from <https://www.databricks.com/blog/efficient-fine-tuning-lora-guide-llms>
- [9] Google Cloud. Evaluating Models. Retrieved March 26, 2024 from <https://cloud.google.com/translate/automl/docs/evaluate>
- [10] Wikipedia. BLEU. Retrieved March 28, 2024 from <https://en.wikipedia.org/wiki/BLEU>
- [11] Ketan Doshi. Foundations of NLP Explained — Bleu Score and WER Metrics. Towards Data Science. Retrieved March 28, 2024 from <https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b>
- [12] Priyanka. Evaluation Metrics in Natural Language Processing — BLEU. Towards Data Science. Retrieved March 28, 2024 from <https://medium.com/@priyankads/evaluation-metrics-in-natural-language-processing-bleu-dc3cfa8faaa5>

# Thank you for your time!

